КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

ПОБУДОВА КОМПЛЕКСНОЇ СИСТЕМИ ЗБЕРІГАННЯ БІНАРНИХ ОБ'ЄКТІВ В ІТ ІНФРАСТРУКТУРІ УНІВЕРСИТЕТУ

Дипломна робота бакалавра
студента 4 року навчання
Спеціальніст: 123 «Комп'ютерна інженерія»
Тимофія ПАРФЕНЮКА

Науковий керівник
канд. техн. наук Олександр БОРЕЦЬКИЙ,
асистент кафедри комп'ютерної інженерії

Рецензент
к.ф.-м.н., Олександр СУДАКОВ,
доцент кафедри медичної радіофізики ФРЕКС

До захисту допускаю: Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____2022 р., протокол № ____

Київ 2022

ЗМІСТ

РЕФЕРАТ

Дипломна робота: 58 с., 27 рис., 5 табл., 16D джерел.

Мета роботи - дослідження і побудова комплексної системи зберігання бінарних об’єктів в IT інфраструктурі університету

Об'єкт дослідження – системи зберігання бінарних об’єктів, високо доступність таких систем, використання систем зберігання бінарних об’єктів в IT інфраструктурі університету.

Дослідження та аналіз проведено за джерелами, що вказані наприкінці, включаючи рекомендації спеціалізованих на цьому компаній, офіційну документацію розробників. Було проведено аналіз різних систем, визначено вимоги та доцільні варіанти.

В першому розділі розглянуто типи даних та принципи зберігання таких даних, а також різні програмні реалізації сховища бінарних об’єктів, проаналізувавши їхні переваги та недоліки, а також визначено вимоги до системи зберігання бінарних об’єктів в IT інфраструктурі університету. У другому розділі проаналізовані вимоги, і згідно з вимогами побудовано комплексне рішення. У третьому розділі описано процес та особливості розгортання системи, а також протестовано роботу системи.

Результатом роботи є визначення вимог до комплексної системи зберігання бінарних об’єктів в IT інфраструктурі університету, визначення архітектури такого рішення, а також побудова даного рішення і розгортання на базі інфраструктури університету.

Ключові слова: СХОВИЩЕ БІНАРНИХ ОБ’ЄКТІВ, СИСТЕМА ЗБЕРІГАННЯ ІНФОРМАЦІЇ, ВИСОКОДОСТУПНЕ СХОВИЩЕ, МЕРЕЖЕВЕ СХОВИЩЕ

ВСТУП

Світ зараз знаходиться в ері інформаційних технологій. Це означає що відбувається велика кількість операцій з різними даними. Такі дані, для того щоб обробляти, потрібно в першу чергу десь зберігати.

На сьогоднішній день перед IT фахівцями стоїть велика задача збереження інформації та даних. З кожним днем кількість користувачів стрімко зростає, а кількість інформації, яку потрібно зберігати, ще швидше зростає. Ті об’єми даних, які ще десятиліття тому вважалися невичерпними, в нас час вже є навіть інколи і недостатніми.

Кожен ресурс, кожен веб-застосунок потребує зберігати свої дані, для подальшого відображення користувачам, або просто для збереження довготривалих даних, або для резервного копіювання важливої інформації. Неправильно побудовані системи збереження даних можуть призвести до страшних наслідків, таких як, втрата важливої інформації, а також отримання чутливої інформації особами, котрі не повинні були мати доступ до даної інформації. Втрата важливої інформації може призвести до серйозних наслідків, навіть аж до закриття певних організацій, або до неможливості таких організацій продовжувати далі свою діяльність.

Отримання таких даних, сторонніми особами може також призвести до наслідків, які будуть впливати як і на організацію, так і на клієнтів, тощо. Користувачі таких систем повинні бути впевнені, що дані, які зберігаються не будуть отримані сторонніми особами.

Перед фахівцями постає задача побудови рішення для оптимального зберігання даних, яка буде задовольняти сучасні вимоги надійності та доступності даних, але в свою чергу буде оптимальна для певних задач. При цьому і вартість такого рішення повинна бути прийнятна для організації, яка її потребує.

При проектуванні та побудові таких рішень, потрібно враховувати велику кількість різних моментів та вимог, щодо доступності та надійності збереження інформації, особливо у випадках, коли не всі користувачі є фахівцями в області інформаційних технологій.

Київський національний університет імені Тараса Шевченка є організацією великого розміру, яка потребує значні ресурси для зберігання різної інформації та даних. Дані, які зберігає університет є дуже чутливими до втрати, а також до таких даних, не повинні отримати доступ сторонні особи. Тому надважливо спроектувати систему зберігання даних, яка буде надійною та безпечною для зберігання важливих даних. Також, не всі співробітники університету, які будуть застосовувати ці системи є IT фахівцями, тому рішення має мати певну простоту використання, але не завдаючи шкоди безпеці.

Тому важливо розглянути і виробити вимоги до системи зберігання бінарних об’єктів в IT інфраструктурі університету, а також спроектувати і розгорнути програмну та апаратну інфраструктуру, яка буде задовольняти поставлені вимоги.

1. МАСШТАБОВАНІ СИСТЕМИ ЗБЕРІГАННЯ ДАНИХ

1.1. Зберігання даних

Зберігання даних означає запис цих даних в оптичному чи електромагнітному вигляді на певний носій, з можливістю потім знайти, прочитати та обробити збережену інформацію.

Зазвичай для збереження даних ми використовуємо засоби операційної системи і зберігаємо інформацію на тих чи інших носіях. Але що в ситуації, коли потрібно робити це віддалено? Що робити, коли потрібно передати інформацію через мережу? Потрібно використовувати мережеві сховища.

Мережеві сховища дозволяють за допомогою мережевого з'єднання передавати дані на сервер збереження, а також, потім, у потрібний час їх отримати для подальшого використання. Це дозволяє зменшувати кількість даних на кінцевих, клієнтських пристроях, а всю потрібну інформацію передавати для збереження і обробки на сервер.

1.1.1 Зберігання структурованої інформації

Зазвичай дані можна поділити на 2 типи: структуровані та неструктуровані. За останні десятиріччя наше розуміння і визначення даних, а також наші уявлення про низ значно змінилися. Зазвичай це спричинене все більшою доступністю новітніх інструментів для збереження, читання, обробки та аналізу неструктурованих даних.

В минулому, неструктуровані дані часто не використовувались, через трудності з їх інтерпретацією та доступом до них. Новітні технології значно полегшили розуміння неструктурованих даних, що надало значний розвиток системам збереження неструктурованих даних.

Структуровані дані зазвичай мають чітко визначену схему для інформації яка там зберігається. Ці дані гарно організовані і точно відформатовані. Зазвичай ці дані існують в форматі реляційних баз даних, це означає, що інформація зберігається в таблицях з зв'язаними рядочками та стовбчиками. Таким чином, структуровані дані акуратно впорядковуються і записуються, що дає такому

методу простоту в пошуку потрібної інформації, а також в її обробці. Різні таблиці можна зв’язати між собою за допомогою спільного стовбчика.

Такі дані зазвичай зберігають у вигляді тексту в реляційних базах даних. Структуровані дані зазвичай мають декілька полів, і у кожного з записів в таблиці є унікальний ідентифікатор, який однозначно характеризує даний запис, для подальшого отримання його з таблиці.

Зазвичай, для отримання доступу для цих даних, а також для маніпуляції ними, використовується мова структурованих запитів SQL. Для продуктивної роботи таких баз даних, в них не прийнято зберігати великі файли, тому що при збільшенні об’єму бази даних, буде зменшуватись швидкодія. Структуровані дані найчастіше зберігаються у вигляді тексту.

Бази даних це системи, які потребують дуже високу швидкодію, високу швидкість доступу до інформації і малу затримку. Такі бази розміщають на високопродуктивних системах, використовують зазвичай швидші диски, такі як твердотільні накопичувачі.

Зі збільшенням користувачів, а також зі збільшенням вимог до надійності та доступності до баз даних, виникло питання масштабування та розподілення таких сховищ. Коли сервіс недоступний користувачам деякий час, це достатньо не приємно, але не смертельно. Якщо ж ми втратимо дані користувача, то це абсолютно не допустимо, і може привести до серйозних наслідків. В основі масштабування баз даних лежить той же принцип, що і в основі масштабування веб-застосунків.

Масштабування може відбуватись горизонтально і вертикально. У основі вертикального масштабування лежить збільшення обчислювальних ресурсів одного і того ж серверу. Наприклад, додавання оперативної пам’яті, дисків або оновлення процесору. Основна перевага цього методу, це його простота. Нема необхідності якось модифікувати програмну інфраструктуру при такому масштабуванні, а управляти одним сервером набагато легше, ніж групою серверів. Таке масштабування також несе свої недоліки, а саме, що ми маємо єдину точку відмови, і у випадку, наприклад диску, ми можемо втратити дані

користувачів. Також, кожен сервер має апаратне обмеження можливих обчислювальних ресурсів.[1]

В наш час, найчастіше використовують саме горизонтальне масштабування. В його основі збільшення продуктивності шляхом розділення даних на певну кількість серверів. Схематично горизонтальне масштабування позначене на рисунку 1.

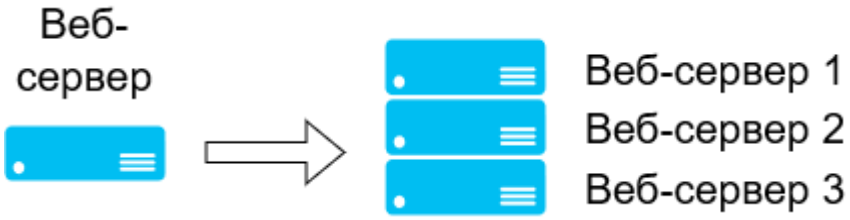

Рисунок 1. Схема горизонтального масштабування

Існує три основних способи горизонтального масштабування:

- Реплікація
- Балансування
- Секціювання
- Сегментування

Реплікація означає копіювання даних між серверами. При використанні такого методу виділяють два типи серверів: мастер і слейв. Мастер використовується для записи та зміни інформації, а слейви для копіювання інформації з мастеру і для читання. Найчастіше використовується один мастер сервер і декілька слейвів, тому що, зазвичай, запитів на запис та зміну інформації набагато менше, ніж запитів на читання. Найголовніша перевага такого методу масштабування – велика кількість копій баз даних. Тому навіть, якщо з деяких причин головний сервер вийде з ладу, будь який інший сервер зможе його замінити. Але такий механізм не зовсім зручний в багатьох випадках, через причини розсинхронізації і затримки передачі даних між серверами. Наприклад, дані можуть вже бути змінені на головному сервері, а на слейв серверах, буде вже не актуальна інформація, що може спантеличити користувача. Тому, найчастіше реплікацію застосовують як засіб для забезпечення відмовостійкості разом з

іншими методами масштабування. Схематично реплікація позначена на рисунку.[1]

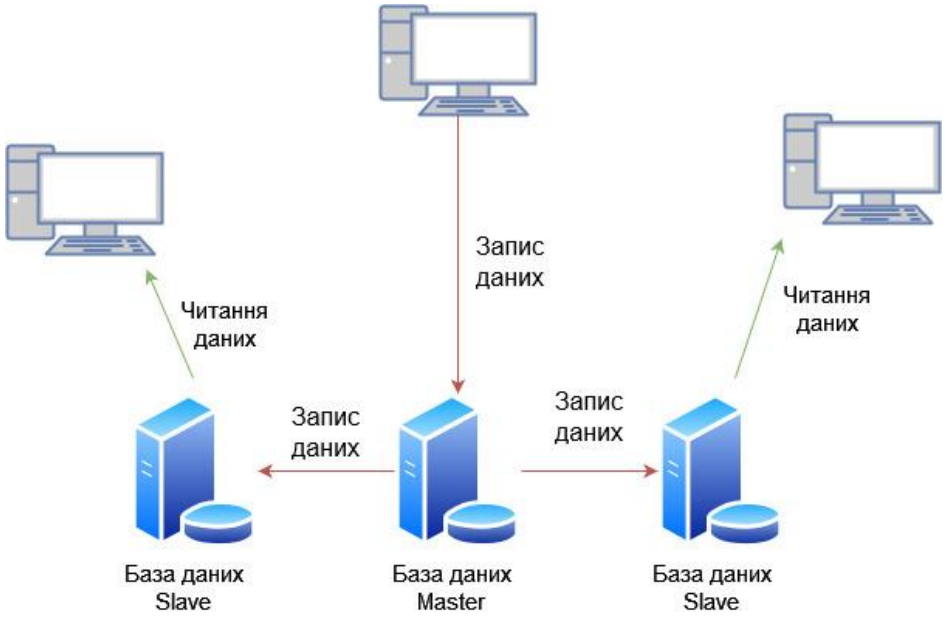

Рисунок 2. Схема реплікації

Ще один з можливих варіантів є можливість використання декількох серверів бази даних, які будуть реплікуватись між собою, а перед ними поставити балансувальник навантаження, який буде розподіляти запити між декількома базами даних. Подібна технологія застосовується для балансування навантаження на веб сторінки. Такий метод також не потребує ніяких змін в програмному забезпеченні, тому що для нього все одно буде вказуватись єдина кінцева точка під’єднання до бази даних.[1]

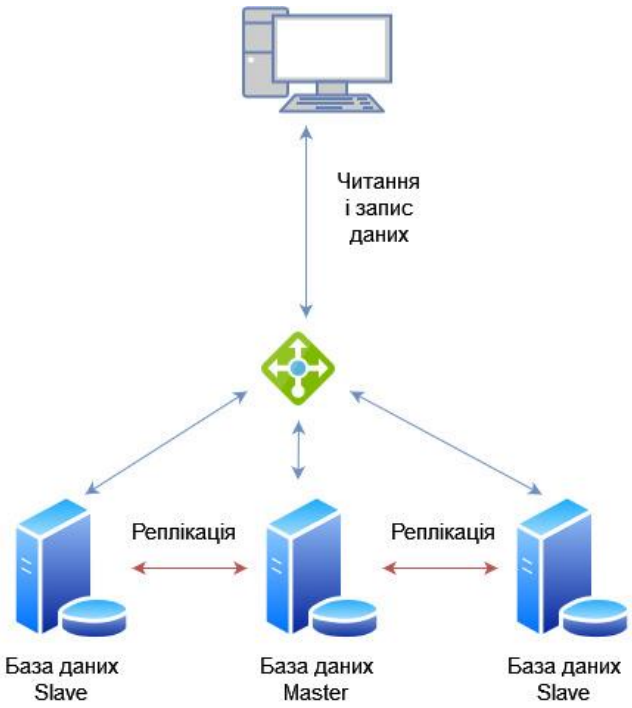

Рисунок 3. Схема балансування

Секціонування заключається в розбитті даних на частини відповідно до деякої ознаки. Наприклад, таблицю можна розбити на дві, за ознакою парності. Причиною для використання такого методу є необхідність підвищення продуктивності. Продуктивність досягається тим, що пошук відбувається не по всій таблиці, а лише по якісь її частині. Ще одна перевага, це можливість швидко видаляти неактуальні фрагменти таблиці.[1]

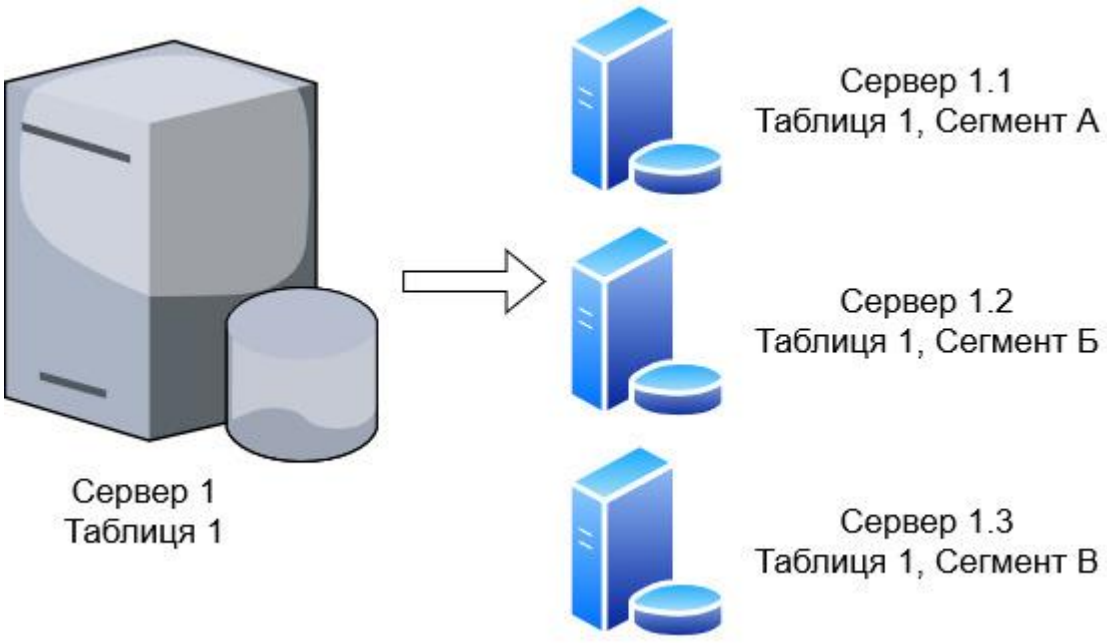

Рисунок 4. Схема секціонування

Сегментація – це принцип проектування бази даних, при якому частини таблиці зберігаються окремо, на різних фізичних серверах. Сегментація є найкращим варіантом для масштабних систем, особливо, якщо її застосовувати у парі з реплікацією. Але цей метод є один з найскладніших в реалізації та організації, так як необхідно враховувати між серверну взаємодію.[1]

1.1.2 Інформація яку не доцільно зберігати в структурованому вигляді

Будь який неструктурований фрагмент даних може бути класифікованим як неструктуровані дані. З кожним днем покращуються кінцеві пристрої користувачів, які знімають кращі відео, фотографії, записують кращий звук. Така інформація є неструктурованою, але дуже важливою. Думаю складно знайти сервіс, який би не оперував файлами. По оцінкам експертів, приблизно 80 відсотків даних до 2025 року, з якими ми будемо стикатися будуть неструктурованими, у вигляді тексту, аудіо, відео чи зображення.

Неструктуровані дані не мають заздалегідь визначену модель даних, що не узгоджує їх з реляційними таблицями і базами даних. Зазвичай неструктуровані дані мають великий об’єм, і бути не просто текстовими даними, а наприклад і бінарними об’єктами. Структурування такої інформації може спотворити її, що унеможливить подальше її відтворення.

Порівняльна характеристика структурованих і неструктурованих даних наведено у таблиці.

Таблиця 1.1.2.1. Порівняння структурованих і неструктурованих даних.

Характеристики	**Структуровані дані**	**Неструктуровані дані**
Характер даних	Зазвичай кількісні	Зазвичай якісні
Модель даних	Попередньо визначена. Після визначення моделі і збереження даних, важко змінити модель	Не мають визначеної схеми; модель даних дуже гнучка. Можна зберігати навіть без чітко визначеної моделі.
Формат даних	Зазвичай текстова	Велика кількість

	інформація, або приведена до текстового вигляду (наприклад, бінарні об’єкти у вигляді тексту закодовані як base64)	всіляких форматів.
Бази даних	Реляційні бази даних	Нереляційні бази даних, файлосховища.
Пошук	Дуже простий пошук по визначених полях і ідентифікаторах	Дуже важко шукати, так як вони не структуровані

Неструктуровані дані дуже складно структурувати. Такі дії можуть призвести до збільшення об’єму даних, а також до падіння продуктивності такої системи. Також, якщо в реляційну базу даних записати такі дані, вона буде мати великий об’єм і буде зменшуватись швидкодія такої системи. Такі дослідження були проведені американським університетом. Якщо в базу даних записувати файли об’ємом, які перевищують 5 мегабайтів, це буде зменшувати продуктивність роботи такої бази даних.

Зазвичай неструктуровані дані мають великий об’єм, наприклад такі як відео, або зображення. Як було сказано раніше, бази даних зазвичай розміщають на швидких дисках, таких як твердотільних накопичувачах. Зберігання такого об’єму інформації на твердотільних накопичувачах, буде недоцільною і дорогою. Тому зазвичай сховища неструктурованих даних розміщують на звичайних жорстких дисках, тому що доступ до неструктурованих даних зазвичай не потребує дуже малих затримок. [6]

Якщо є необхідність зберігати дані в початкових форматах, не змінюючи їх, для подальшого аналізу, то краще всього для цього підходять репозиторії сховища, або ще як їх називають, озера даних.

Методи масштабування, які були розглянуті вищі дуже не доцільно використовувати для неструктурованих даних. Реплікація таких сховищ буде

займати великий час і тратити надто багато ресурсів і оновлення всіх залежних баз даних буде займати дуже багато часу. Це буде негативно впливати на користувачів. Отже, зовсім недоцільно використовувати методи масштабування і підвищення доступності систем збереження структурованих даних, для сховищ неструктурованих даних.

1.2. Бінарні об’єкти даних

Бінарні об’єкти – це масиви двійкових даних або набір двійкових даних, що зберігаються як єдиний об’єкт. Зазвичай у вигляді бінарних об’єктів зберігається двійковий виконуваний код, а також зображення, аудіо, відео та інші мультимедійні об’єкти.

Такий тип даних і визначення були введені для опису даних, які ще тоді були не визначеними в традиційних на той час базах даних. Це було через те, що в той час такі файли були надто великими, щоб зберігати їх в полях баз даних. З часом застосування такого типу даних набуло більшого поширення, місце на дисках збереження стало дешевше.

З часом більшість баз даних почали підтримувати збереження бінарних об’єктів, а також з’явилася підтримка маніпуляції бінарними об’єктами в деяких мовах програмування, таких як JavaScript. Але, як було сказано раніше, бінарні об’єкти дуже не доцільно зберігати в сховищах структурованих даних.

Давайте спробуємо більш детально розібратися в проблемі збереження бінарних об’єктів в реляційних базах даних, і яка проблема структуризації таких даних. Що ж таке структурування інформації? Структурування – це виділення важливих елементів в інформаційних повідомленнях і встановлення зв’язків між ними. Коли ми розглядаємо бінарні об’єкти, то їх ми не можемо розбити на якісь важливі елементи, тому що це просто послідовність бітів та байтів, тому такі дані зазвичай не структуруються, а просто записуються в поле бази даних, відповідно до схеми. [6]

Також для усіх файлових форматів для бази даних це буде єдиний тип, бінарний об’єкт (BLOB), тому це може призвести до плутаниці в даних для користувача і розробника, що дуже погана практика.

Про проблему збереження бінарних об’єкті у реляційній базі даних написав також Білл Карвін, автор книги «SQL Antipatterns». Це американський розробник, інженер, який отримав ступінь в Університеті Каліфорнії, і більше 20 років пропрацював в полі інформаційних технологій. Він описав, що збереження бінарних об’єктів у базі даних, дуже погана практика. По перше, зазвичай в базах даних, якщо не використовується об’єкт FILESTREAM, то максимальний розмір файу не повинен перевищувати 2 гігабайти. Якщо ж потрібно більше, то потрібно придумувати «милиці» через що, будуть порушені деякі основні принципи проектування даних.

По друге, розмір бази даних збільшиться. Зазвичай в ній зберігається дуже критична інформація, і тому резервне копіювання налаштоване чи не кожен день, а інколи і частіше. Якщо ж там будуть зберігатися бінарні об’єкти, вартість і ресурси які потрібні для резервного копіювання бази даних зростуть. Тут же виникає і проблема з реплікацією, яка вже була описана вище. При великому розмірі даних, реплікація буде займати дуже великий час, через що можуть виникати проблеми з актуальністю реплік.

По третє, якщо робити вибірку з будь яким стовпчиком, який містить BLOB, то ми завжди будем звертатися до диску, в той час коли без стовпчиків з бінарними об’єктами ми можемо отримувати дані напряму з ОЗП. Особливо це потрібно для баз даних з високою пропускною здатністю.

Якщо ж говорити про хмарні сервіси, то послуги збереження даних в базах даних набагато дорожчі, ніж ті ж самі файлові сховища.

1.3. Особливості зберігання бінарних об'єктів в IT інфраструктурі університету

Якщо взяти поставлену задачу, а саме інтеграцію даного рішення в IT інфраструктуру університету, ми зіткнемося з певними особливостями.

В Україні відсутні нормальні механізми оплати хмарних ресурсів державними структурами. Також, якщо ж все таки знайти спосіб оплати, виникає ще проблема в тому, що будь-які витрати повинні прораховуватись та вноситись в

бюджет на наступний рік. Якщо ж взяти університет, то складно спрогнозувати витрати на хмарну інфраструктуру.

Виходячи з цього потрібно буде розглядати сервіс, який буде базуватися на власній, існуючій інфраструктурі.

Даний сервіс буде інтегрований з багатьма іншими сервісами, тому його доступність буде безпосередньо впливати на доступність інших сервісів. Це означає, що даний сервіс повинен мати високу доступність.

Кількість користувачів цього сервісу буде великою, і об’єм трафіку буде також достатньо високий.

Якщо ж розглядати саме КНУ імені Тараса Шевченка, та підсумувати все написане вище, то до сховища бінарних об’єктів можна виділити ще такі вимоги:

- Зберігання великої кількості «архівної інформації»
- Кількість GET та PUT запитів, які можуть постійно змінюватись
- Достатньо великий обсяг трафіку
- Можливість інтеграції з готовими програмними рішеннями
- Продукт простий до використання
- Висока доступність
- Висока продуктивність

1.4. Зберігання бінарних об’єктів та існуючі рішення сховищ

Розглянувши антипатерни зберігання бінарних об’єктів, залишається питання, то все таки, як краще зберігати бінарні об’єкти. Ми знаємо що всі файли, що включає в себе і бінарні об’єкти, зберігаються у файловій системі. Отже, для нашої задачі, а саме для системи зберігання об’єктів, яка буде доступна через мережу потрібна певна абстракція, яка дозволить отримувати дані. Це і називається об’єктним сховищем.

Об’єктне сховище – це додатковий шар абстракції над файловою системою і хостом, який дозволяє працювати з файлами, отримувати доступ, зберігати, через певні мережеві інструменти взаємодії, наприклад API. [2]

Зберігати дані, наприклад, певного програмного забезпечення або веб ресурсу, можна у різний спосіб, наприклад зберігати просто на диску серверу, або

зберігати у базі даних (неактуальність цього рішення було розглянуто вище). Але чи є таке рішення оптимальним? Звісно ні. Часто є певні нефункціональні вимоги, які б нам хотілося б задовольнити та реалізувати, такі як масштабованість, простота підтримка та гнучкість. У випадках, коли зберігання файлів побудовано на роботі з локальною файловою системою сервера, таке рішення як масштабування продукту буде відбуватися дуже проблематично.

І от на допомогу приходять саме об’єктні сховища. Об’єктне сховище це спосіб зберігати дані та гнучко отримувати до них доступ як до об’єктів (файлів). У цьому контексті об’єкт – це файл та набір метаданих про нього.

Якщо підсумувати, то перевага використання файлового сховища у тому, що ми отримуємо певну абстракцію, над самою файловою системою, що для кінцевого користувача, або для продукту, ПЗ, які зв’язуються з файловим сховищем, їм не потрібно знати і розуміти внутрішню структуру і організацію цього файлового сховища. Ми можемо масштабувати, змінювати організацію, але при цьому не змінюючи кінцеву точку підключення (end-point), для користувача ніяких змін відбуватися не буде.

На даний момент існують багато різних варіантів організації файлового сховища або сховища об’єктів. Розвиток технологій і розширення можливостей і збільшення варіантів різних можливих застосувань спричинило це.

Ці сервіси відрізняються між собою. Деякі з них мають графічні інтерфейси, прості в налаштуванні та експлуатації. Інші доволі складні, не мають графічних інтерфейсів, потребують деяких модифікацій програмного забезпечення, для використання. Але навіть не зважаючи на це, більшість з компаній, які розробляють програмне забезпечення і працюють з даними використовують одне з таких рішень в свої інфраструктурі.

Також більшість з таких сховищ об'єктів мають реалізації в мовах програмування для більш простого використання, наприклад існує бібліотека для підтримки NFS для C#/.NET.

Питання забезпечення надійного зберігання, безпеки та доступності критично важливий даних мають першочергову важливість. При розгляді варіанту

зберігання даних у ідеальних файлових сховищах є кілька фундаментальних вимог:

- Надійність. Дані повинні зберігатися надмірно. В ідеалі вони повинні бути розподілені між кількома об’єктами та кількома пристроями у межах кожного з об’єктів. Стихійні лиха, людський фактор або механічні несправності не повинні призводити до втрати даних
- Доступність. Усі дані мають бути доступними у разі потреби.
- Безпека. В ідеалі всі дані повинні шифруватись – як при зберіганні, так і при передачі.

Взагалі всі можливі рішення можна розділити на два типи:

- Cloud-based – застосування хмарних ресурсів. Не відповідаємо за апаратну та програмну інфраструктуру. Користуємось, як готовим продуктом. Платимо по факту використання.
- Self-hosted – застосування власних ресурсів. Організовуємо та адмініструємо апаратну та програмну інфраструктуру.

1.4.1. Сховища бінарних об’єктів у хмарній інфраструктурі

Хмарне сховище – це модель хмарних обчислень, що передбачає зберігання даних в Інтернеті за допомогою постачальника хмарних обчислювальних ресурсів, який надає сховище даних як сервіс, та забезпечує керування ним. Хмарне сховище надається на вимогу у необхідному обсязі, оплачується за фактом використання та позбавляє необхідності купувати власну інфраструктуру для зберігання даних та керувати нею. Це забезпечує гнучкість, глобальну масштабованість та надійність. Дані будуть доступні у будь-який час та в будь-якому місці. [7]

Хмарне сховище купується у стороннього постачальника хмарних сервісів, який володіє ресурсами сховища даних, керує ним та надає доступ до них через інтернет з оплатою за фактом використання. Постачальники хмарних сховищ відповідають за стан ресурсів, безпеку та надійність, забезпечуючи доступність даних для програм клієнтів по всьому світу.

Програми отримують доступ до хмарного сховища через традиційні протоколи зберігання даних або через API. Багато постачальників також пропонують додаткові послуги, призначені для захисту, збору та аналізу даних у величезних масштабах, а також управління ними.

Зберігання даних у хмарі має великі переваги:

- Сукупна вартість. Завдяки хмарному сховищу не потрібно купувати обладнання, виділяти ресурси для сховища та витрачати кошти на інфраструктуру «прозапас». В хмарних сховищах можна додавати або видаляти ресурси на вимогу, швидко змінювати продуктивність та термін зберігання. І при цьому платиться тільки за використовувані ресурси. Дані, які використовуються рідше, можна автоматично переміщати на більш економні плани зберігання за правилами, які можна розробити і модернізовувати. Це дозволяє забезпечити економію за великих обсягів.
- Час до розгортання. Коли продукт готовий до запуску, або виникає вимога в додаткових системах зберігання, інфраструктура не повинна бути стримуючим фактором. Хмарне сховище дозволяє IT-спеціалістам швидко виділяти необхідний простір для зберігання даних, саме тоді, коли це потрібно.
- Управління інформацією. Централізоване сховище у хмарі створює величезні можливості нових прикладів використання. Використовуючи політики управління життєвим циклом у хмарному сховищі, можна вирішувати важливі завдання, пов'язані з управлінням інформацією, включаючи автоматичний розподіл за рівнями або блокування даних з метою дотримання певних вимог чи правил.

В даній роботі буде розглянуто 3 сервіси сховищ об'єктів, від найпопулярніших постачальників послуг хмарних обчислень:

- AWS S3
- Google Cloud Storage
- Microsoft Blobs Storage

В загальному це дуже схожі сервіси. Вони виконують однакові функції, хоча кожен з них має певні особливості.

1.4.1.1. Хмарний сервіс AWS S3

Amazon Simple Storage Service – це сервіс зберігання об’єктів, розроблений компанією Amazon і пропонується для використання у Amazon Web Services (AWS). Цей сервіс має одні з кращих в галузі показники продуктивності, масштабованості, доступності та безпеки даних. Клієнти будь-якої величини та з будь-якої промислової галузі можуть зберігати та захищати необхідний обсяг даних для практично будь-якого варіанту використання, наприклад, озера даних, хмарні або мобільні додатки. Вигідні тарифи, класи сховищ та прості у використанні інструменти адміністрування дозволяють оптимізувати витрати, організувати дані та точно налаштувати обмеження доступу відповідно до потреб бізнесу чи законодавчих вимог.[3]

На даний момент, хмарний сервіс AWS має 80 зон доступності (80 availability zones) у 25 географічних регіонах світу. При використанні сервісу S3 дозволяється налаштовувати реплікацію між різними зонами доступності, для пришвидшення доступу до даних, в залежності від географічного розположення.

Даний сервіс має декілька варіантів роботи з ним: через консоль, встановлюючи спеціальне програмне забезпечення, а також через браузер, веб версія. Права доступу можна дуже тонко налаштувати, описавши їх у json форматі, що також дає можливість спільного редагування і резервного копіювання через системи контролю версій.

Дана система має доступність рівну 99,999999999% (119 секунд недоступності) в рік. .

Із недоліків, одна з найвищих вартостей зберігання гігабайту на ринку. Також тарифікується не тільки кількість інформації, яка зберігається, а також трафік і кількість GET та PUT запитів.

Аналіз даного рішення з поставленими вимогами до системи зберігання об’єктів наведено в таблиці 1.4.1.1.1.

Таблиця 1.4.1.1.1. Відповідність до вимог бінарного сховища.

Критерій	Відповідність
Зберігання великої кількості «архівної інформації»	Даний сервіс дозволяє недорого зберігати велику кількість архівної інформації, яка буде надана за запитом. Сервіс пропонує високу доступність таких даних.
Кількість GET та PUT запитів, які можуть постійно змінюватись	Постійна зміна GET та PUT унеможливлює спрогнозувати витрати на зберігання даних у цьому сервісі.
Достатньо великий обсяг трафіку	Великий обсяг трафіку на завантаження файлів буде підвищувати вартість зберігання даних.
Можливість інтеграції з готовими програмними рішеннями	Дане рішення має велику кількість бібліотек для інтеграції з різними системами та програмними рішеннями.
Продукт простий до використання	Налаштування і експлуатація може відбуватись тільки фахівцями з відповідними кваліфікаціями.
Висока доступність	Дана система має доступність рівну 99,999999999% (119 секунд недоступності) в рік.
Висока продуктивність	Дана система має високу продуктивність.

1.4.1.2. Хмарний сервіс *Google Cloud Storage*

Google Cloud Storage або GCS – це сервіс зберігання об’єктів, розроблений компанією Google. Даний сервіс працює в Google Cloud Platform (GCP). Має достатньо широкі можливості для реплікації і постійно зростаючий список можливих локацій для реплікації даних. [3]

Даний сервіс має спеціально розроблену утиліту, яка з легкістю допомагає перенести дані з локальних пристроїв, або з іншого сховища об’єктів.

Доступ до сховища і управління може відбуватися через веб консоль, через консоль локальної машини, через встановлене програмне забезпечення, а також через браузер.

Із переваг, даний сервіс має доступність 99.999999999% (119 секунд недоступності) в рік..

Із недоліків, даний сервіс має найдорожчу ціну на трафік для скачування файлів, а також має більш складне налаштування прав доступу до бакетів.

Аналіз даного рішення з поставленими вимогами до системи зберігання об’єктів наведено в таблиці 1.4.1.2.1

Таблиця 1.4.1.2.1. Відповідність до вимог бінарного сховища.

Критерій	Відповідність
Зберігання великої кількості «архівної інформації»	Даний сервіс дозволяє зберігати велику кількість архівної інформації. Сервіс пропонує високу доступність таких даних.
Кількість GET та PUT запитів, які можуть постійно змінюватись	Вартість запитів найвища з усіх рішень. Постійна зміна GET та PUT унеможливлює спрогнозувати витрати на зберігання даних у цьому сервісі.
Достатньо великий обсяг трафіку	Вартість вхідного та вихідного трафіку найдорожча з усіх розглянутих хмарних рішень. Великий обсяг трафіку на завантаження файлів буде підвищувати вартість зберігання даних.
Можливість інтеграції з готовими програмними рішеннями	Дане рішення має велику кількість бібліотек для інтеграції з різними системами та програмними рішеннями.
Продукт простий до використання	Налаштування і експлуатація може відбуватись тільки фахівцями з відповідними кваліфікаціями.
Висока доступність	Дана система має доступність рівну 99,999999999% (119 секунд недоступності) в рік.
Висока продуктивність	Дана система має високу продуктивність.

1.4.1.3. Хмарний сервіс *Microsoft Blobs Storage*

Сховище BLOB-об'єктів Azure — це рішення корпорації Майкрософт для зберігання об'єктів у хмарі. Сховище BLOB-об'єктів оптимізовано для зберігання величезних обсягів неструктурованих даних.

Користувачі або клієнтські програми можуть отримати доступ до об'єктів у сховищі BLOB-об'єктів через HTTP/HTTPS з будь-якої точки світу. До об'єктів у сховищі BLOB-об'єктів можна звертатися через REST API служби сховища Azure, Azure PowerShell, Azure CLI або клієнтську бібліотеку служби сховища Azure. [3]

Основна особливість, що у сховищі BLOB-об'єктів пропонується три типи ресурсів:

- обліковий запис зберігання;
- контейнер в обліковому записі зберігання;
- бінарний об'єкт у контейнері.

Переваги:

- Найдешевший з усіх досліджених сховищ бінарних об’єктів.
- Зрозуміла і локалізована документація

Недоліком є певна особливість організації даних в цьому сховищі бінарних об’єктів.

Аналіз даного рішення з поставленими вимогами до системи зберігання об’єктів наведено в таблиці 1.4.1.3.1

Таблиця 1.4.1.3.1. Відповідність до вимог бінарного сховища.

Критерій	Відповідність
Зберігання великої кількості «архівної інформації»	Даний сервіс дозволяє зберігати велику кількість архівної інформації. Сервіс пропонує високу доступність таких даних.
Кількість GET та PUT запитів, які можуть постійно змінюватись	Вартість запитів найнижча з усіх рішень. Постійна зміна GET та PUT унеможливлює спрогнозувати витрати на зберігання даних у цьому сервісі.
Достатньо великий обсяг трафіку	Вартість вхідного та вихідного трафіку найдешевша з усіх розглянутих хмарних рішень. Великий обсяг трафіку на завантаження файлів буде підвищувати вартість зберігання даних.

Можливість інтеграції з готовими програмними рішеннями	Дане рішення має велику кількість бібліотек для інтеграції з різними системами та програмними рішеннями.
Продукт простий до використання	Налаштування і експлуатація може відбуватись тільки фахівцями з відповідними кваліфікаціями.
Висока доступність	Дана система має доступність рівну 99,999999999% (119 секунд недоступності) в рік.
Висока продуктивність	Дана система має високу продуктивність.

1.4.2. Програмні рішення сховищ на власній інфраструктурі

Інший варіант використання і застосування сховища об'єктів, це розгортання на власній інфраструктурі. Причини для такого рішення можуть бути різні. Одна з них, це коли ми використовуємо хмарні сховища, ми не можемо бути на 100 відсотків впевнені в конфіденційності даних, які там зберігаються.

Ще одна причина може бути існування власної інфраструктури, що буде дешевше ніж платити за використання хмарних сховищ.

Зазвичай такі рішення застосовують для навчання або тестування програмного забезпечення, щоб мінімізувати витрати, але також такі рішення можна застосовувати і для продакшн рішень.

Такі рішення застосовуються і компаніями, які не мають можливості, в силу якихось особливостей, оплачувати хмарні сховища. Або згідно з наказів або вказівок, коли потрібно щоб дані не зберігалися за межами організації, або країни, тощо.[9]

Такий варіант має і свої недоліки. Підтримка інфраструктури, як програмної, так і апаратної лягає на плечі адміністраторів. Також, дуже складно дотримуватись такої ж доступності, як і у постачальників хмарних обчислень, тому що не завжди організації можуть повністю контролювати наявність електрики та телекомунікаційних з’єднань, які залежать не від них. [10]

Також, адміністраторам потрібно прорахувати різні потенційні проблеми і вирішити їх. Потрібно щоб і рішення на власній інфраструктурі могло

масштабуватись, а також було розподілене, був налаштований моніторинг, з повідомленнями у разі проблем, а також щоб підтримувалась потрібна доступність і надійність даних.

Всі рішення можна поділити на 3 категорії:

- Мережеві файлові системи (Ceph, GlusterFS)
- Мережеві служби (NFS, Samba, ISCSI)
- Мережеві файлові сховища. (Minio)

1.4.2.1. Мережева файлова система Ceph

Ceph – система збереження об’єктів, що забезпечує як файловий, так і блочний інтерфейси доступу. Може використовуватись на системах, що складаються з однієї машини, так і з тисяч вузлів, наприклад, як система збереження даних в компанії Yahoo, яка побудована на Ceph і призначена для збереження даних розміром у кілька сотень петабайт. [8]

Така система дуже просто масштабується, виконує різні функції, забезпечуючи реплікацію даних, а також розподіл навантаження, що гарантує високу доступність і надійність.

При виході будь-якого диску, вузла з ладу Ceph не тільки забезпечить збереження даних, але й сам відновить втрачені копії на інших вузлах, до тих пір, поки вузли або диски, що вийшли з ладу не замінять на працюючі. При цьому така дія відбувається без простоїв і прозоро для клієнтів.

Основним недоліком цієї системи є важкість в встановленні та експлуатації. Навіть для достатньо освідчених адміністраторів встановлення даної системи може викликати певні складності.

Зазвичай доступ відбувається за допомогою використання блочних пристроїв, або підключення як мережевого диску. Також доступний веб-панель керування для відображення важливої інформації.

Дане рішення не має обмежень на кількість GET та PUT запитів. Обмеження на кількість трафіку, а також продуктивність залежить від апаратної та мережевої інфраструктури.

Доступ через API відбувається через «милиці», тому зазвичай його не використовують. Це ускладнєю інтеграцію з іншими програмними рішеннями.

1.4.2.2. Мережева файлова система GlusterFS

GlusterFS – це розподілена, паралельна, лінійно масштабована файлова система з можливістю запобігання та захисту від збоїв. GlusterFS працює в просторі користувача, тому не вимагає підтримки з боку ядра операційної системи і працює поверх існуючих файлових систем. На відміну від Ceph, не потребує окремого серверу для зберігання метаданих.[8]

Доступ можна отрмати використовуючи технологію FUSE, або через NFS. Доступу через API не передбачено. Можливість інтеграції з іншими продуктами дуже складна.

Дане рішення не має обмежень на кількість GET та PUT запитів. Обмеження на кількість трафіку, а також продуктивність залежить від апаратної та мережевої інфраструктури.

Налаштування та експлуатація даної системи відбувається фахівцями з інформаційних технологій, так як для звичайного користувача дана система є складною.

1.4.2.3. Мережева служба NFS

Використання такої файлової системи має складності у масштабуванні. Розширення простіру буде потребувати переформатування. Доступність та надійність зберігання даних залежить від налаштування апаратної інфраструктури і не залежить від даної файлової системи.

В залежності від версії, пропускна здатність відрізняється, але є доволі низкою. Кількість запитів і кількість трафіку, в тому числі і одночасного регулюється апаратною та мережевою інфраструктурою.

Дана файлова система складна в налаштуванні і експлуатації і потребує певних навичок у користувача.

Доступ відбувається тільки через протокол NFS. Доступ через API не передбачений. Контроль доступу дуже складний і неочевидний в налаштуванні.

1.4.2.4. Мережева служба Samba

Samba – пакет програм, які дозволяють звертатися до мережних дисків та принтерів на різних операційних системах за протоколом SMB/CIFS. Має клієнтську та серверну частини.

В своїй реалізації дуже схожа на NFS. Кількість запитів і кількість трафіку, в тому числі і одночасного регулюється апаратною та мережевою інфраструктурою

Доступ відбувається тільки через протокол Samba. Доступ через API не передбачений. Контроль доступу дуже складний і неочевидний в налаштуванні. Також даний програмний проект має доволі багато вразливостей, які постійно експлуатуються.

1.4.2.5. Мережеве файлове сховище Minio

MiniIO – це популярне програмне забезпечення з відкритим кодом для розподіленого зберігання об'єктів, сумісне з S3. Він відомий своєю високою продуктивністю. [15]

Ви можете використовувати MinIO для простого веб-додатку до великих робочих навантажень розподілу даних для програм аналітики та машинного навчання.

Плюси використання:

- Стандартне сховище плоских файлів
- Розподіл даних у кількох вузлах
- Аварійне відновлення
- Аналітика даних

Minio записує дані та метадані як об'єкт. Це усуває залежність від наявності додаткової бази даних або програмного забезпечення для зберігання метаданих та підвищення продуктивності.[13]

Переваги використання Minio:

- Minio здатний читати/писати зі швидкістю ~ 170 ГБ/с.
- Масштабованість – використовуйте кластеризацію та масштабуйте при необхідності

- Cloud-native
- Захист даних із використанням методу Erasure code
- Підтримується множинне шифрування, включаючи AES-CBC, AES-256-GCM, ChaCha20

Сервіс Minio дуже схожий в реалізації на сервіс S3 від AWS. Має веб інтерфейс для доступу і завантаження даних, а також веб інтерфейс для налаштування, контролю доступу і т.д. Доволі простий для експулатації користувачем. Має зрозумілий інтерфейс.

Політики можна прописувати через json файли, що дає змогу для колективної взаємодії з файлами конфігурації.

Має велику кількість бібліотек, що дозволяє інтеграцію з різними програмними продуктами.

Даний продукт може бути розгорнуте в високодоступному та високопродуктивній конфігурації, якщо воно буде використовувати платформу оркестрації контейнерів Kubernetes з підключенням до Kubernetes розподіленого сховища зберігання даних.

1.5. Опис рішення для університетської інфраструктури

Отже, вище були розглянуті різні варіанти сховищ бінарних об’єктів як і у хмарній інфраструктурі так і у власній. Зазвичай, якщо розглядати рішення для якоїсь приватної компанії, найкращим рішенням є застосування готових рішень, як програмних, так і інфраструктурних. Тому, якби якась приватна бізнесова компанія розглядала б варіанти збереження неструктурованих даних, найкращим би для них було б застосування готового рішення від постачальників хмарних обчислень, таких як Amazon S3, Google Cloud Storage або Microsoft Blobs Storage.

Враховуючи всі перечислені вище особливості, а також те, що університет вже має певну інфраструктуру, так як мережеву і сервери, то найкращий варіант буде розгортати програмне рішення на власній інфраструктурі.

Якщо враховувати що потрібно буде це рішення просто інтегрувати з готовими програмними продуктами, а також простоту використання для деяких кінцевих користувачів, то краще всього застосовувати S3 сумісні рішення.

Найкращим з розглянутих вище програмних рішень, для розгортання в IT інфраструктурі університету є MinIO.

Найкращим рішенням є використання орекстратору контейнерів, а саме Google Kubernetes, використовуючи всі переваги які надає цей сервіс.

Так як контейнери зберігають дані тільки протягом свого рантайму, нам потрібно передбачити реалізацію постійного сховища для Kubernetes. Найкращим варіантом буде використання Ceph на окремому серверу і під'єднання як блочного пристрою.

2. АРХІТЕКТУРА КОМПЛЕКСНОЇ СИСТЕМИ ЗБЕРІГАННЯ БІНАРНИХ ОБ'ЄКТІВ В IT ІНФРАСТРУКТУРІ УНІВЕРСИТЕТУ.

Для забезпечення роботи мережевих сховищ для об'єктів необхідно реалізувати апаратну та програмну інфраструктуру.

Зазвичай, більшість додатків чи сервісі простіше всього налаштувати, як певний сервіс, чи програма, яка встановлюється в операційній системі. Ми просто беремо фізичинй сервер, або віртуальну машину, куди встановлюємо операційну систему, і налаштовуємо той чи інший сервіс.

Деякі дистрибутиви мають певні особливості, налаштування тих чи інших сервісів. Тому потрібно враховувати платформо залежність, і складність переносу сервісів на іншу ОС.

Також при використанні такого підходу масштабування, реплікація та перенос у випадку незвичайної ситуації кінцевої точки буде викликати певні складності.

Велику популярність зараз має застосування контейнеризації, коли застосунок, разом із усіма залежностями запаковується у контейнер, і може бути перенесеними між різними машинами і навіть різними операційними системами. Але за звичайних умов, контейнеризований застосунок мало чим буде відрізнятися від просто встановленого і налаштованого на локальній системі. Тому в такій ситуації є сенс використовувати систему оркестрації контейнерів.[11]

Оркестрація дозволяє створювати інформаційні системи з безлічі контейнерів, кожен із яких відповідає лише одне певне завдання, а спілкування здійснюється через мережні порти і загальні каталоги. За потреби кожен такий контейнер можна замінити іншим, що дозволяє, наприклад, швидко перейти на іншу версію бази даних за потреби.

Існують різні платформи для оркестрації контейнерів. Вони дозволяють реалізувати зручні та ефективні засоби розгортання контейнерних систем, побудови єдиної централізованої консолі для застосування політик управління.

Апаратна та програмна інфраструктура є однією з найважливіших частин будь якого сервісу. Правильність побудови та планування інфраструктури може в майбутньому спростити і передбачити багато проблем.

Враховуючи всі сучасні практики нам потрібно скласти вимоги, передбачивши можливі сценарії різних неординарних ситуацій.

Вимоги до інфраструктури:

- Можливість масштабування
- Попередження про проблеми
- Передплачування надзвичайних ситуацій
- Автоматичне опрацювання надзвичайних ситуацій
- Можливість легкого переносу інфраструктури
- Можливість розподілення інфраструктури
- Висока доступність

Враховуючи всі дані умови нами були підібрані наступні варіанти інфраструктури.

Загалом, нам потрібне рішення, в якому ми зможемо розгорнути високодоступну та високопродуктивну конфігурацію Minio. Комплексна схема зображена на рисунку 5.

Рисунок 5. Комплексна схема архітектури сховища бінарних об’єктів

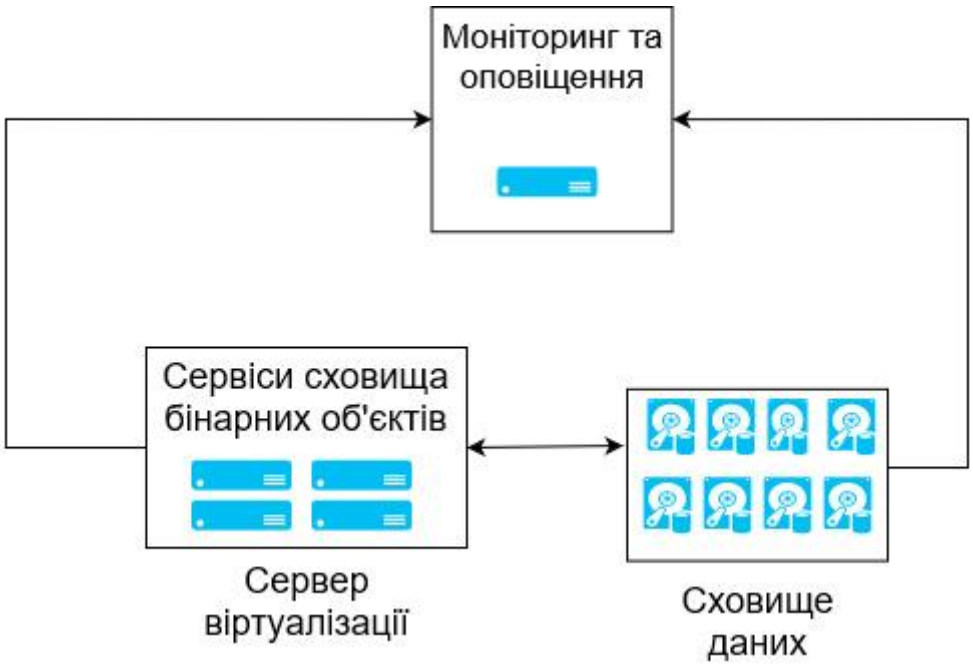

Рисунок 6. Схема архітектури сховища бінарних об’єктів

2.1. Операційна система для розгортання системи зберігання об’єктів

Загалом існує два основних варіанта, коли підходити до вибору серверної операційної системи: Windows Server та Linux. Сервери на Linux-подібних ОС вважаються безвідмовними і надійними. Тому все більше користувачів вибирають Linux дистрибутиви.

Найбільш популярними серверними дистрибутивами ОС Linux є CentOS, Debian та Ubuntu. Для IT інфраструктури університету найкращим рішенням є використання CentOS, так як більшість інфраструктури використовує цю операційну систему. Це дозволить полегшити інтеграцію нашого рішення.

Враховуючи описані вище особливості даної операційної системи та дивлячись на вимоги до університетського сховища бінарних об’єктів, можна

сказати, що дана система є надійною основою для забезпечення безперебійної роботи усіх сервісів.

Також завдяки високої популярності і великі підтримці від компанії RedHat дана операційна система є доволі безпечна. Постійні оновлення безпеки випускаються і різні вразливості закриваються.

Отже, використання даної операційної системи для побудови рішення задовільнить наступні вимоги: безпека даних, надійність, висока продуктивність.

Також було сказано, що вся існуюча інфраструктура університету використовує дану операційну систему. Тому дане рішення буде найкраще для побудови нашого сховища.

2.2. Гіпервізор для розгортання системи зберігання об’єктів

Для оптимального використання всіх ресурсів серверу, а також для забезпечення ізольованості окремих сервісів варто застосовувати віртуалізацію. Дане рішення допомагає створити окрему віртуальну машину для кожного сервісу, яка буде мати власне віртуальне «залізо». Різні сервіси будуть взаємодіяти тут через мережу. Також буде можливість переносу однієї віртуальної машини з одного сервера віртуалізації на інший, що дозовлить розподіляти інфраструктуру.

Існує декілька типів гіпервізорів, гіпервізор-ОС, це такі, які встановлюються як операційна система (ESXI), або так, які встановлюються на існуючу операційну систему (VMWare, Hyper-V, KVM). VmWare та Hyper-V в більшості своєму інсталюються на операційну систему Windows. В нашому випадку, так як використовується операційна система Linux, нам потрібен гіпервізор, який інсталюється поверх операційної системи найкращим варіантом є KVM.

Оскільки KVM є стандартним модулем ядра Linux, він отримує від ядра всі належні перевапги (робота з пам'яттю, планувальник тощо). А відповідно, зрештою, всі ці переваги дістаються і гостям (бо гості працюють на гіпервізорі, що працює на/в ядрі ОС Linux).

KVM дуже швидкий, та його самого недостатньо для запуску віртуальної ОС, т.к. для цього потрібна емуляція пристроїв введення та виведення. Для їх

емуляції (процесор, диски, мережа, відео, PCI, USB, серійні порти тощо) KVM використовує QEMU.

В даному рішенні сховища бінаринх об’єктів, для кожного сервісу (групи сервісів) буде забезпечена окрема віртуальна машина, для ізоляції їх один від одного, що забезпечить додатковий прошарок безпеки та допоможе уникнути конфліктів різних сервісів.

Отже, дане рішення буде допомагати нам реалізувати такі вимоги університетського сховища даних, як оптимальність використання ресурсів, та безпека, так як кожен сервіс буде ізольований один від одного.

2.3. Розподілене файлове сховище для розгортання системи зберігання об’єктів

Для реалізації сховища об'єктів нам потрібно розгорнути розподілене файлове сховище. Варіантами є GlusterFS та Ceph.

Ceph дозволяє створити цілий кластер, який буде працювати як одне мережеве сховище. При цьому вузли кластера можуть бути розділені між собою навіть географічно.

У випадку бажання, в майбутньому даний сервіс можна буде розширити, додавши нові диски або вузли кластера, для кращої безперебійності.

GlusterFS має схожі характеристики як і Ceph, але на даний момент розробники GlusterFS не підртимують сервісів підключення до Kubernetes, що може призвести до проблем у майбутньому. Тому на даному етапі найкращим варіантом буде використання Ceph.[16]

Даний сервіс буде підключатися до Kubernetes кластеру для створення постійно сховища для сервісів, які потрібні для побудови університетської системи сховища бінарних об’єктів.

Отже, дане рішення буде допомагати нам реалізувати такі вимоги університетського сховища даних, як розподілення даних, бо дані зможуть зберігатися в різних місцях, а також надійність сховища, тому що у випадку якоїсь непередбачуваної ситуації, дані можна буде відновити. Також, дане

рішення дає можливість до подальшого масштабування у випадку, коли це буде потрібно.

2.4. Система оркестрації контейнерів для розгортання системи зберігання об’єктів

В даній інфраструктурі ядро нашого сховища, а саме сервіс Minio та всі супутні сервіси, такі як Minio Console, будуть запускатися в Docker контейнерах в системі орекстарції контейнерів.. Це дозволить мати ще додаткову ланку стабільності.

Також, у випадку підвищеного навантаження на даний сервіс, буде можливість масштабувати сервіс, додаючи нові екземпляри сервісу і масштабуючи навантаження між ними.

Також у майбутньому для забезпечення більшої стабільності є можливість територіального розділення інфраструктури, а саме запуск ще декількох вузлів кластеру які будуть знаходитися в іншій територіальній позиції.

Існує два найпопулярніших системи орекстрації контейнерів: Docker Swarm та Kubernetes.

Недоліком Docker Swarm є те, що в нього меньша кількість підтримуваних операційни систем, а також прив’язка тільки до Docker API. Також має гіршу автоматизацію у випадку непередбачуваних обставин.

Kubernetes розробляється компанією Google, тому має гарну підтримку, та більше різних варіантів автоматизації та обробки виключних ситуаці. Тому найкращим рішенням буде застосування оркестратору Kuberenetes. [11]

Дане рішення дозволить задовольнити такі вимоги університетського сховища даних, як надійність інфраструктури, яка буде постійно тестуватись, можливість подальшого масштабування і розподілення інфраструктури, а також висока продуктивність, яка буде досягатись автоматичним масштабуванням екземплярів сервісу при підвищеному навантаженні.

2.5. Забезпечення високої доступності системи зберігання об’єктів

Щоб забезпечити високу доступність системи зберігання об'єктів, потрібно вчасно взнавати про стан системи кожної компоненти. Для цього потрбіно

використовувати моніторинг системи. Існує багато різних систем моніторингу: Zabbix, Prometheus.

Основною перевагою Prometehus є його відкритість, що дозволяє подальшу модифікацію.

На кожну віртуальну та фізичну машину, яка буде використовуватись для побудови сховища, буде встановлено сервіс NodeExporter, який буде збирати метрики з машин, а сам Prometheus буде їх приймати і зберігати.

Grafana буде застосовуватись для візуалізації отриманих метрик та їх відображення. Це буде дозволяти виявляти аномалії в поведінці інфраструктури.

Для відправляння повідомлень про надзвичайні ситуації, такі як зникнення серверу з мережі, через якісь негаразди, або про високу нагрузку, буде повідомляти сервіс AlertManager. Він буде відправляти повідомлення в канал в копроративний месенджер Mattermost. [13]

Для цих сервісів буде розгорнута окрема віртуальна машина.

Дане рішення дозволить задовольнити такі вимоги університетського сховища даних, як надійність інфраструктури, та отримання повідомлень у надзвичайних ситуаціях.

2.6. Програмна реалізація комплексної системи зберігання об’єктів

Даний сервіс, можна сказати, буде ядром нашого всього сховища.

MiniIO – це популярне програмне забезпечення з відкритим кодом для розподіленого зберігання об'єктів, сумісне з S3. Він відомий своєю високою продуктивністю. [15]

Ви можете використовувати MinIO для простого веб-додатку до великих робочих навантажень розподілу даних для програм аналітики та машинного навчання.

Даний сервіс буде налаштований для запускання в Kubernetes кластері. Для розгортання буде використаний Helm Chart.

Також, поряд з ним буде використовуватись Minio Console, веб сервіс для налаштування та адміністрування сервісу Minio. Даний сервіс буде розгортатись за допомогою написаного маніфест файлу для деплойменту.

Як сховище постійних даних, буде пі'єднано блочний пристрій з Ceph. Даний блочний пристрій буде розміром 45 терабайт.

Дане рішення дозволить задовольнити такі вимоги університетського сховища даних, як простота в використанні і налаштувані. Також даний сервіс дуже легко інтегрувати з існуючими проектами, застосунками та іншими рішеннями, які застосовуються в університетській інфраструктурі.

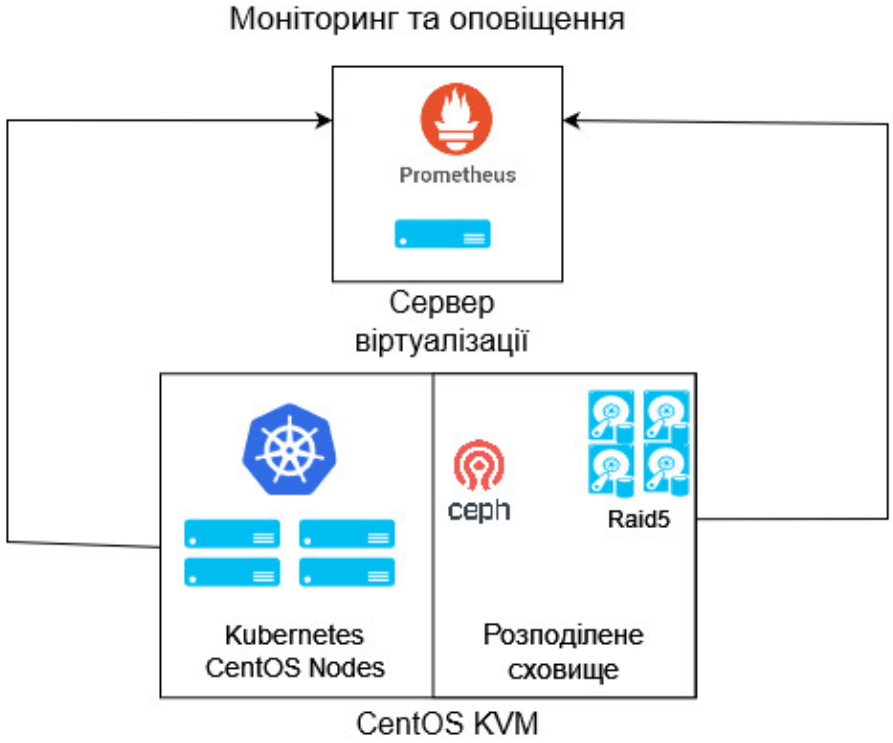

Рисунок 7. Схема архітектурного рішення сховища бінарних об’єктів

3. ПОБУДОВА ІНФРАСТРУКТУРИ І АРХІТЕКТУРНІ РІШЕННЯ

3.1. Опис апаратної інфраструктури

Як було описано вище, дана система буде складатися з багатьох сервісів.

Основою стане фізична машина, а саме сервер Supermicro. Даний сервер має процесор з 48 ядрами, а також 128 гігабайт оперативної пам’яті.

Як сховище для інформації будуть використовуватись 7 дисків на 10 терабайт, а для метаданих і для операційних системи віртуальних машин будуть використовуватись 2 SSD диски по 500 гігабайт.

Даний сервер буде мати гігабітне підключення до мережі інтернет. Для доступу віртуальних машин до інтернету, а також для обміну даними між собою буде створений віртуальний міст.

Отже, ми будемо мати один фізичний сервер, де кожен сервіс буде розгорнутий у окремій віртуальній малині. Це дозволить нам досягнути високу доступність, тому що у випадку збою однієї віртуальної машини, це не буде мати впливу на інші віртуальні машини. Також ізоляція сервісів дозволяє підвищити безпеку, адже у випадку доступу до однієї віртуальної машини, до інших доступу не буде.

Завдяки використанню віртуальних машин буде можливість оптимально використовувати ресурси і масштабуватись у майбутньому.

В майбутньому все розглянута нами архітектура дозволить масштабуватись і розширитись і навіть до декількох фізичних серверів.

Всі сервіси будуть розгортатись на основі операційної системи сімейства CentOS.

Середовище оркестрації контейнерів Kubernetes, яка буде основою для розгортання сервісу, а також розподілене мережеве сховище Ceph будуть розгорнуті у вигляді віртуальних машин на сервері Supermicro. Для моніторингу, всі сервіси будуть підв’язані до системи моніторингу Інформаційно-обчислювального центру університету.

Для автоматизованого налаштування і розгортання будуть використовуватись Ansible скрипти та плейбуки. Це дозволить мати на всіх

віртуальних машинах однакову конфігурацію, а також відслідковувати зміни в інфраструктурі.

Всі сервери розміщені в серверній інформаційно-обчислювального центру університету. Для віддаленого доступу до фізичного серверу налаштований IPMI, який дозволить віддалено налаштовувати фізичний сервер.

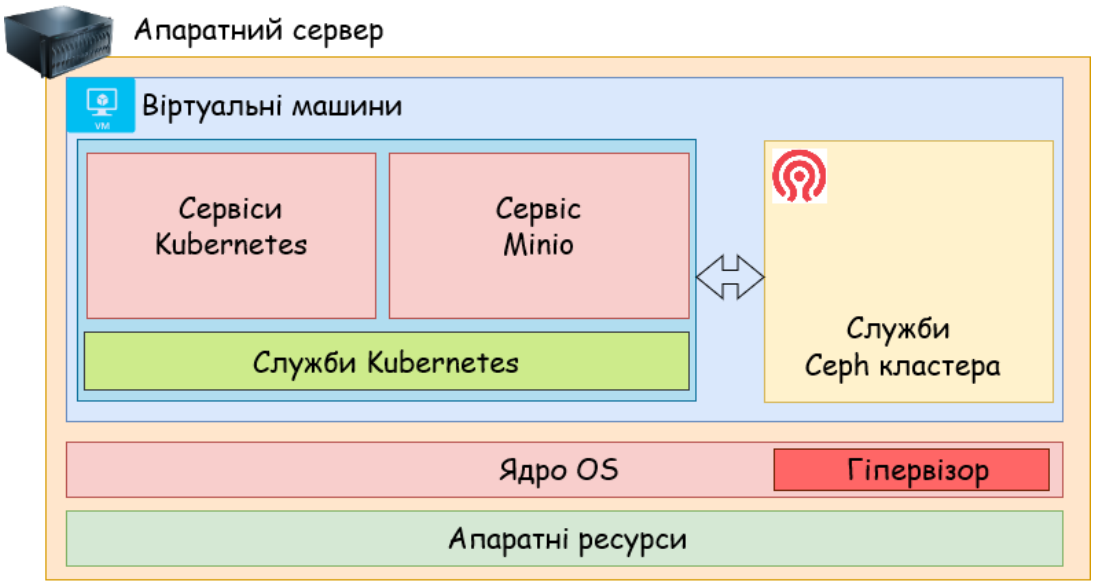

Рисунок 8. Концептуальна схема рішення

3.2. Підготовка та налаштування інфраструктури

Для розгортання комплексної системи зберігання бінарних об'єктів потрібно провести налаштування різних компонент інфраструктури. На даному етапі потрібно підготувати «платформу» для розгортання різних компонент системи.

На даному етапі буде налаштований сервер віртуалізації, налаштовані мережеві та інші служби, розгорнуте розподілене файлове сховище Ceph і налаштований оркестратор контейнерів Kubernetes.

3.2.1. Налаштування серверу віртуалізації

Для початку потрібно налаштувати сервер віртуалізації. Потрібно встановити і налаштувати операційну систему. На даному сервері будемо використовувати операційну систему CentOS 8.

В плані безпеки, доступ до даного серверу відбувається через протокол SSH з аунтефікацією по ключу, а доступ по паролю відключений. Це дозволить суттєво зменшити безпекові ризики, а саме підбір паролю.

Даний сервер буде знаходитись в мережі 10.10.108.0/24. Для управління мережею буде використовуватись демон systemd-networkd. Для віртуальних машин буде використовуватись мережа 10.10.109.0/24. Для створення моста для доступу в мережу віртуальних машин в каталозі /etc/systemd/network/ потрібно створити файл 10-esoc.network з даним вмістом:

```
[Match]
Name=esoc

[Network]
DHCP=no

[Bridge]
STP=no
```

Для встановлення сервера віртуалізації потрібно встановити наступні пакети:

```
dnf install libvirt libvirt-daemon-driver-qemu.x86_64 qemu-kvm.x86_64
```

Для створення та управління віртуальними машинами можна використовувати командний рядок. Але доцільніше використовувати програмне забезпечення virt-manager, що являє собою графічний застосунок, який може підключатись до серверу віртуалізацій за протоколом SSH, і дозволяє створювати, редагувати і підключатись до віртуальних машин. Приклад даного застосунку наведений на Рисунку 8.

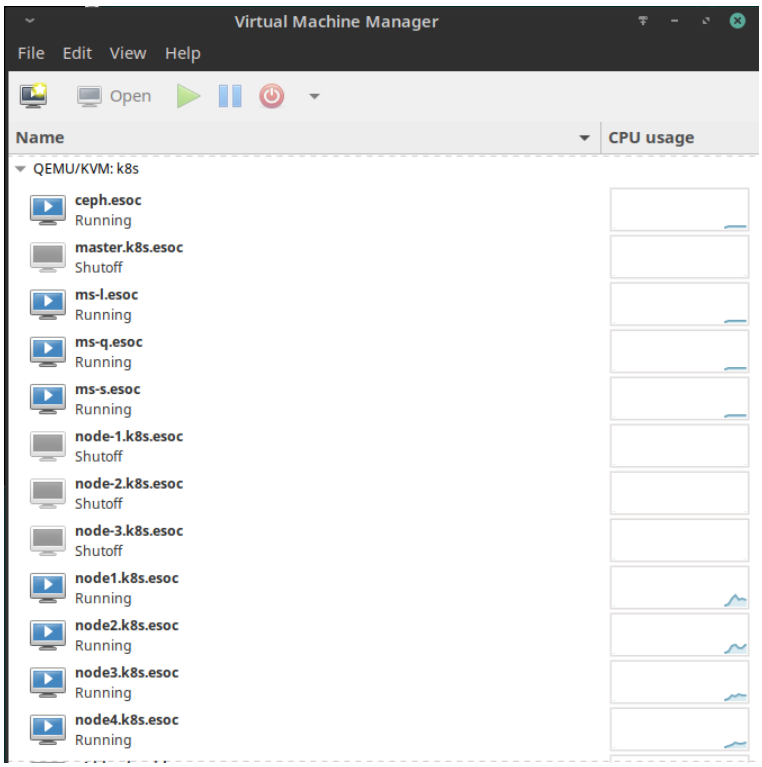

Рисунок 9. Вікно застосунку virt-manager

3.2.2. Налаштування LVM

Для збереження даних будуть використовуватись віртуальні диски, в створених групах, на реальних дисках. Загалом буде дві групи: для віртуальних машин і постійне сховище. Один з SSD дисків, буде доданий в групу для віртуальних машин. Всі жорсткі диски будуть додані в групу для зберігання інформації, а SSD диск буде використовуватись для кешування і метаданих.

Приклад створення групи віртуальних дисків для збереження даних віртуальних машин.

```
# pvcreate /dev/nvme0n1p4
# vgcreate cube-vm /dev/nvme0n1p4
```

Для збереження великої кількості постійних даних використовуватимуться жорсткі диски. Для підвищення надійності зберігання інформації на цих носіях, їх об’єднано в RAID 5 масив. Це дозволить, навіть у випадку втрати одного з дисків, відновити всю інформацію. LVM дозволяє програмно об’єднати ці носії в один масив.

Для підвищення швидкодії, бажано додати швидкий накпоичувач для кешування та зберігання метадати. При створенні логічного пристрою LVM у нас є така можливість.

Створимо розділ для кешу в групі cube-data.

```
# lvcreate -n ssd-cache cube-data -L 454.8G /dev/nvme1n1
```

Створимо розділ для метаданих кешу в групі cube-data.

```
# lvcreate -l 100%FREE -n ssd-cache-metadata cube-data /dev/nvme1n1
```

Переконвертуємо і об’єднаємо розіли для кешу і метаданих.

```
# lvconvert --type cache-pool --poolmetadata cube-data/ssd-cache-metadata cube-data/ssd-cache
```

Переконвертуємо і об’єднаємо розіли для кешу і метаданих.

```
# lvcreate -l 100%FREE -n 45TB cube-data /dev/md0
```

```
# lvconvert --type cache --cachemode writeback --cachepool cube-data/ssd-cache cube-data/45TB
```

В результаті в нас створено 5 дисків для віртуальних машин і один диск для збереження даних, який підєднаємо до розподіленого сховища Ceph.

```
EL8 HW KNU ESOC:cube [root ~]# lvs | grep 'cube-data\|node'| grep -v node-
  45TB               cube-data Cwi-aoC---  45.46t [ssd-cache_cpool] [45TB_corig] 53.77  0.07           0.01
  node1.k8s.esoc     cube-vm   -wi-ao----  20.00g
  node2.k8s.esoc     cube-vm   -wi-ao----  20.00g
  node3.k8s.esoc     cube-vm   -wi-ao----  20.00g
  node4.k8s.esoc     cube-vm   -wi-ao----  20.00g
  node5.k8s.esoc     cube-vm   -wi-a-----  20.00g
EL8 HW KNU ESOC:cube [root ~]#
```

Рисунок 10. Виведення логічних дисків LVM

3.2.3. Встановлення віртуальних машин

Встановлення операційної системи не містить ніяких особливостей. Розбивка диску відбувається таким чином: 10 гігабайт під розділ /, та 10 гігабайт під розділ /var.

Також на процесі встановлення операційної системи потрібно налаштувати мережевий адаптер, а також налаштувати пароль.

Для налаштування системи використовується Ansible скрипт. Це дозволяє швидко привести віртуальну машину до потрібного стану і одночасно

налаштовувати велику кількість віртуальних машин, застосовуючі зміни, які нам потрібні. Скрипт наведений за посиланням https://gl.icc.knu.ua/esoc/kubernetes/k8s-vm-init/-/blob/main/roles/centos/tasks/main.yml. Також за посиланням можна знайти пам'ятку для адміністратора застосування цього скрипта.

Основна задача скрипта полягає у налаштуванні SSH серверу, щоб дозволяти доступ виключно за ключем, додавання ключів адміністраторів, оновлення системи та встновлення необхідних утиліт.

3.2.4. Налаштування моніторингу

Усі агенти системи моніторингу інтегровані з центральною системою моніторингу. Ядром цієї системи є програмне забезпечення Prometheus, а для візуалізації отриманих даних використовується веб сервіс Grafana.

Встановлення агентів на віртуальні машини відбувається на етапі ініціальної конфігурації, і встановлюються за допомогою скрипта, який наведений за посиланням https://gl.icc.knu.ua/esoc/kubernetes/k8s-vm-init/-/blob/main/roles/node_exporter_install/tasks/main.yml.

Цей комплекс сервісів дозволяє отримувати дані з віртуальних машин, обробляти їх і відображати у вигляді різних метрик та графіків, які зрозумілі для людей. Вигляд основної сторінки сервісу Grafana показана на рисунку 10:

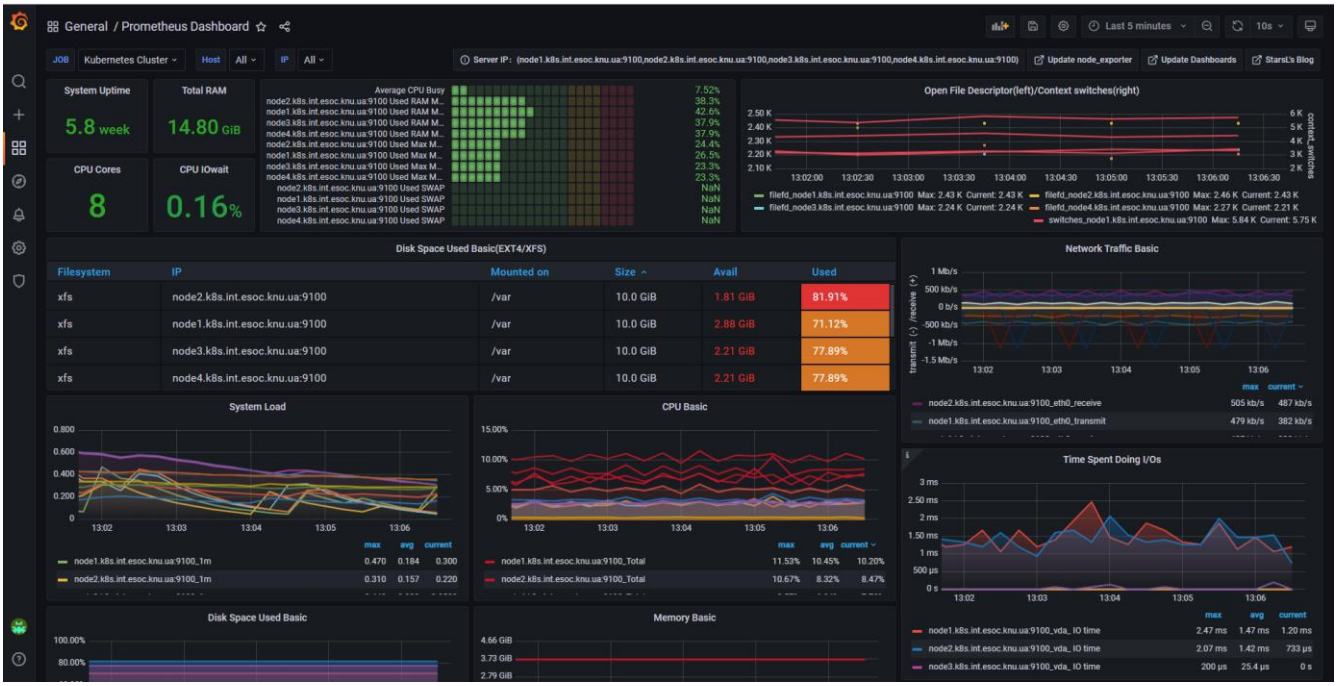

Рисунок 11. Вікно сервісу Grafana та метрики з віртуальних машин

Основна задача моніторингу, виявляти аномальну поведінку і вчасно реагувати на цю поведінку. Такими ситуаціями може бути високе навантаження на основні компоненти, або ж навіть непередбачуване вимкнення віртуальної машини або серверу віртуалізації. Для вчасного реагування на такі ситуації використовуються алерти (повідомлення).

Для відправлення повідомлень використовується сервіс AlertManager, який аналізує зібрані метрики за певними правилами і відправляє повідомлення у вибраний месенджер. На рисунку 11 можна побачити приклад повідомлення про недоступність вузла Kubernetes.

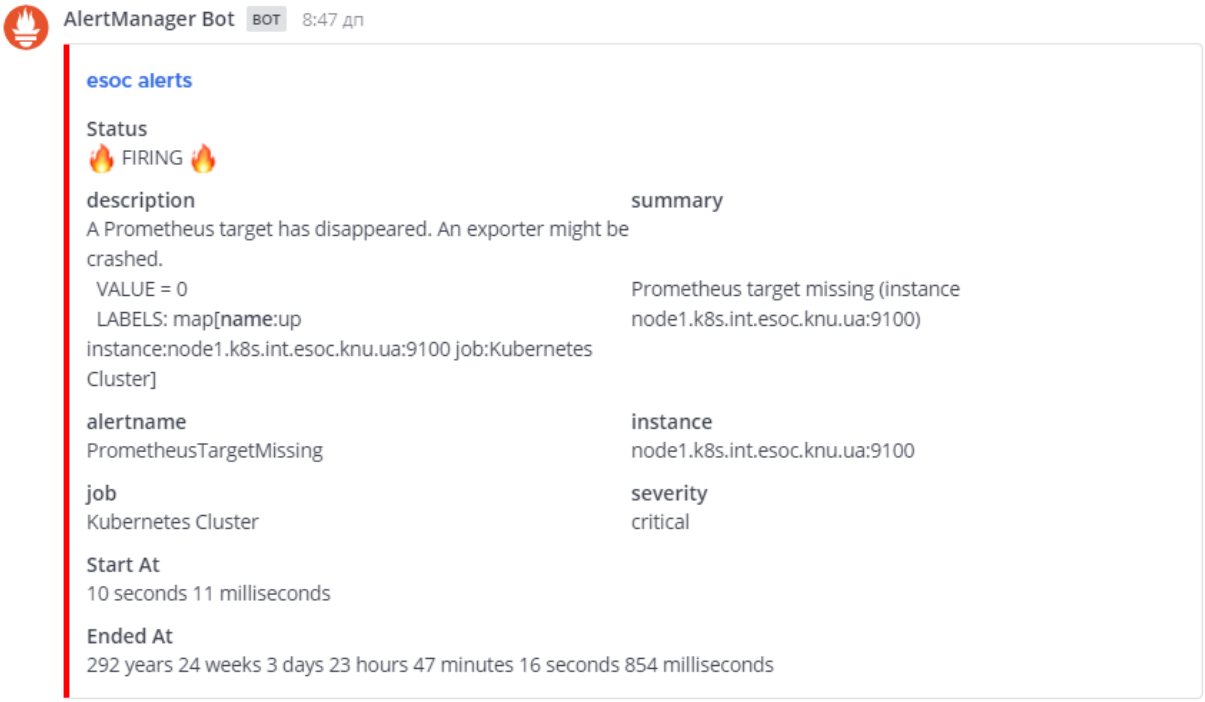

Рисунок 12. Приклад алерту AlertManager у застосунку Mattermost

3.2.5. Інсталяція та налаштування Ceph

Сервіс інсталюється на віртуальну машину з CentOS 8. Даний сервіс буде мати конфігурацію single node cluster. Особливістю є те, що потрібно додати ще один диск, який буде зберігати дані з цього сервісу.

Для зберігання інформації буде використовуватись 7 дисків на 10 терабайт. Для підвищення надійності зберігання інформації на цих носіях, їх об’єднано в RAID 5 масив. Це дозволить, навіть у випадку втрати одного з дисків, відновити всю інформацію. LVM дозволяє програмно об’єднати ці носії в один масив. Для

підвищення швидкодії, буде додано один SSD диск, який буде зберігати кеш та метадані. До віртуальної машини буде підключено як звичайний віртуальний диск на 45 терабайт.

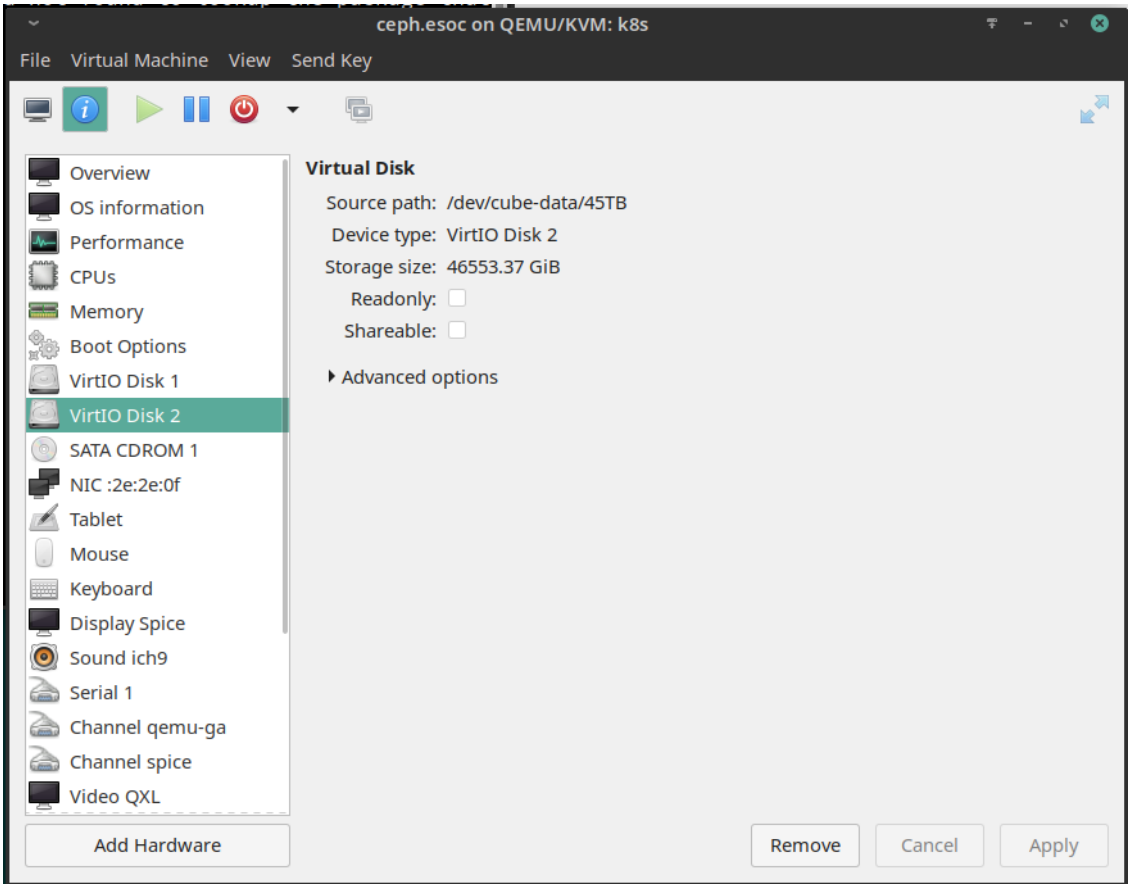

Рисунок 13. Додавання диску на 45 терабайт до віртуальної машини

Даний сервіс має інсталятор, а саме Ansible скрипт, який дозволяє з легкістю інсталювати даний сервіс. Репозиторій знаходиться за посиланням: https://github.com/ceph/ceph-ansible. Задача адміністратора внести зміни у файли змінних, в залежності від потрібної конфігурації кластера, а також вказати адреси хостів, які будуть складати даний кластер у хост файл. Основні змінні, які потрібно внести для коректної роботи сервісу, наведені за посиланням https://gl.icc.knu.ua/esoc/ceph-octopus-deploy/-/blob/main/group_vars/all.yml. Далі запускається скрипт і отримуємо готовий кластер Ceph. Інсталяція може займати деякий час, і може завершуватись невдачею. Інколи потрібно перезапустити виконання скрипта декілька разів, для успішного встановлення. [12]

З особливостей, диск, який буде вказаний, як сховище для кластеру, повинен бути без розмітки і таблиць, тому що це може унеможливити виконання скрипта, і буде викликати помилки.

Для полегшення адміністрування даного сервісу існує панель керування, яка відображає основну інформацію і довзоляє внести певні зміни і зконфігурувати наш кластер.

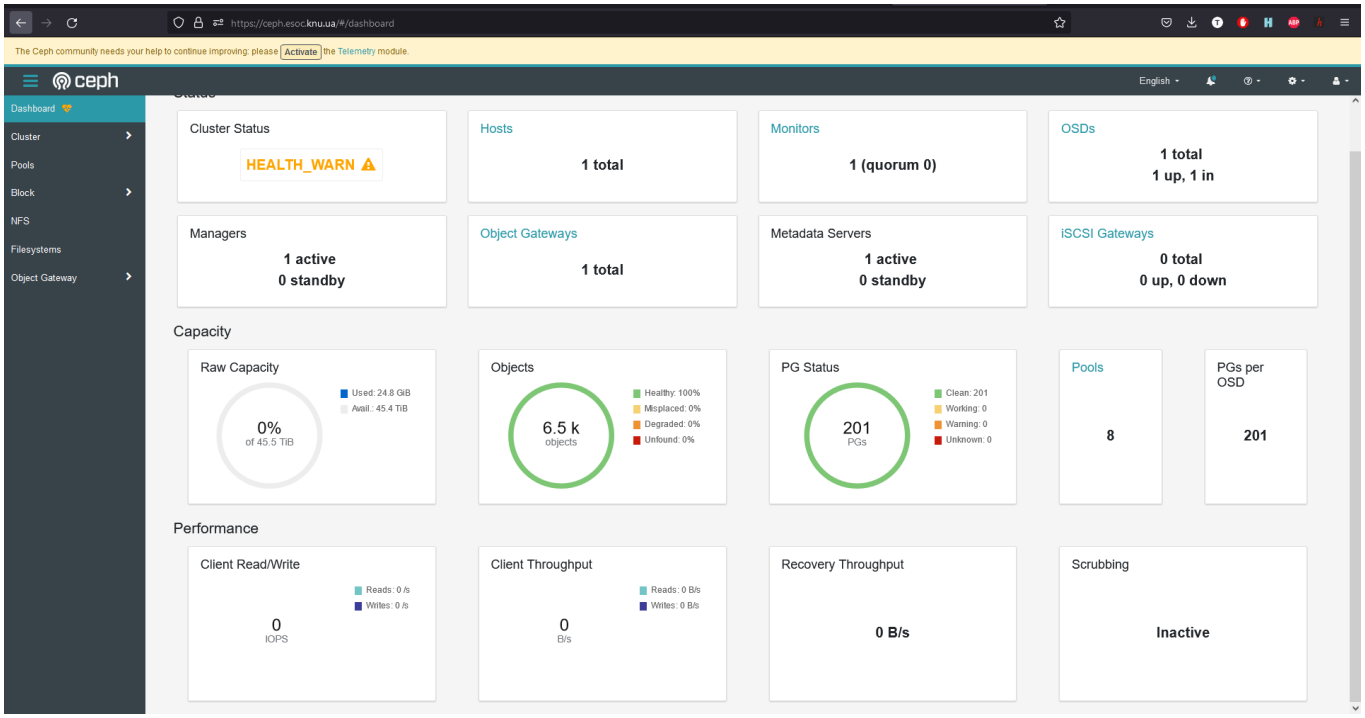

Рисунок 14. Панель керування Ceph

3.2.6. Інсталяція та налаштування Kubernetes

Розгортання і налаштування кластеру скаладється з таких частин:

- Інсталяція клстеру
- Налаштування MetalLB
- Налаштування NginxIngress
- Налаштування Kubernetes Dahsboard
- Інтеграція Kubernetes з Ceph

Розгортання Kubernetes кластеру буде відбуватися за допомогою Kubespray. Це Ansible скрипт, перевага якого в тому, що в ньому мається багато різних налаштувань, які дозволяють розгорнути саме ту конфігурацію кластера, яка потрібна.[11]

В даному випадку розгортається кластер, який має 2 керуючі вузли. Це дозволяє забезпечити додаткову стабілність у разі виходу з ладу одного з вузлів кластеру.

В даному випадку встановлюється особлива версія Kubernetes, а саме v1.21.3, тому що наступні сервіси мають певну несумісніть з провізіонером томів в Ceph кластері.

Всі залежності встановлюються на етапі конфігурування системи. Також на етап встановлення потрібно вимкнути фаєрвол, про що сказано в документації до Kubespray.

Основні змінні та інвенторі файли вказані за посиланням https://gl.icc.knu.ua/esoc/kubernetes/kubespray-esoc/-/blob/master/inventory/mycluster/hosts.yaml. Встановлення займає десь біля години, після чого ми отримуємо кластер готовий до роботи.

Для роботи деяких сервісів необхідно отримувати певну IP адресу. Для цього потрібно інсталювати і налаштувати балансувальник, який буде бронювати певну адресу, і всі запити на цю адресу перенаправляти на певний сервіс в Kubernetes кластері.

Найкраще для цього підходить MetalLB. Даний сервіс резервує адреси із заданого діапазону адрес. Встановлення складає собою виконня двох маніфест файлів, які можна отримати на офіційні сторінці.

Далі потрібно створити конфігураційний файл з діапазонами адрес, які будуть видаватись для сервісів.

Приклад конфігураційного маніфесту за посиланням https://gl.icc.knu.ua/esoc/kubernetes/metallb/-/blob/main/configMap.yaml

Нижче можна побачити приклад сервісу, який отримав адресу від балансувальника (стовбчик External-ip):

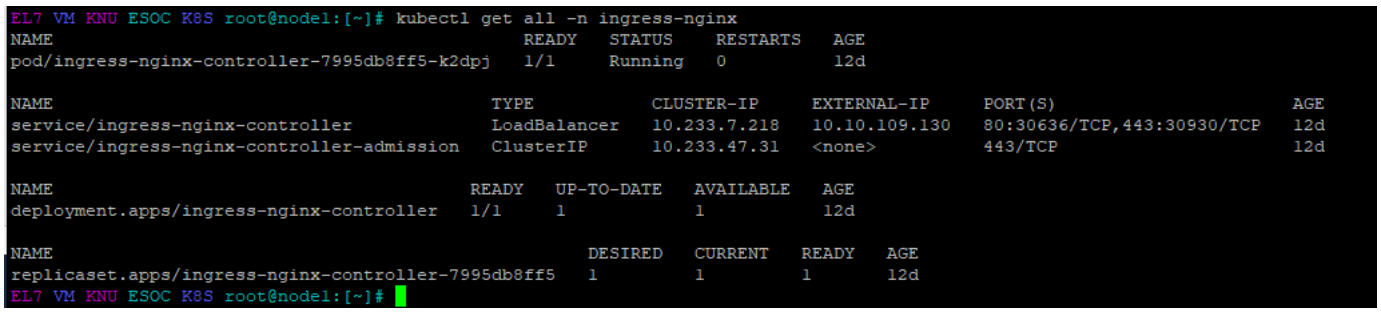

Рисунок 15. Виведення сервісів nginx-ingress у Kubernetes

Задачею ingress-controller є проксування запитів в залежності від адреси чи домену, який був заданий до певного сервісу. Для даної задачі найкраще підходить Nginx ingress-controller. Встановлення являє собою виконання helm сценарію з заданими змінними.[11]

Для полегшення моніторингу і адміністрування кластеру, був налаштований Kubernetes Dashboard, який дозволяє дивитися всі наявні сервіси, а також редагувати маніфест файли. Для встановлення потрібно виконати маніфест файл. Основна особливість в ньому, що для доступу використовується тип сервісу NodePort, це означає що при запиті на певний порт будь якого вузла кластеру, запити будуть перенаправлятись на даний сервіс. Це дозволило налаштувати балансування навантаження.

Ще потрібно створити сервісний аккаунт і роль, а щоб отримати ключ доступу до цього сервісу. За допомогою наступної команди можна отримати ключ доступу для новоствореного акаунту, який надасть доступ до панелі:

```
kubectl -n kubernetes-dashboard get secret $(kubectl -n kubernetes-dashboard get sa/admin-user -o jsonpath="{.secrets[0].name}") -o go-template="{{.data.token | base64decode}}"
```

Сервіс надає вичерпну інформацію про стан різних компонент системи, дозволяє керувати ними, змінювати та видаляти сервіси і тощо.

Для додаткового захисту, доступ до даної панелі дозволений тільки з внутріншної мережі університету, або з зазделегіть заданих адрес, домашніх адрес адміністраторів.

Вигляд панелі зображений на рисунку 15:

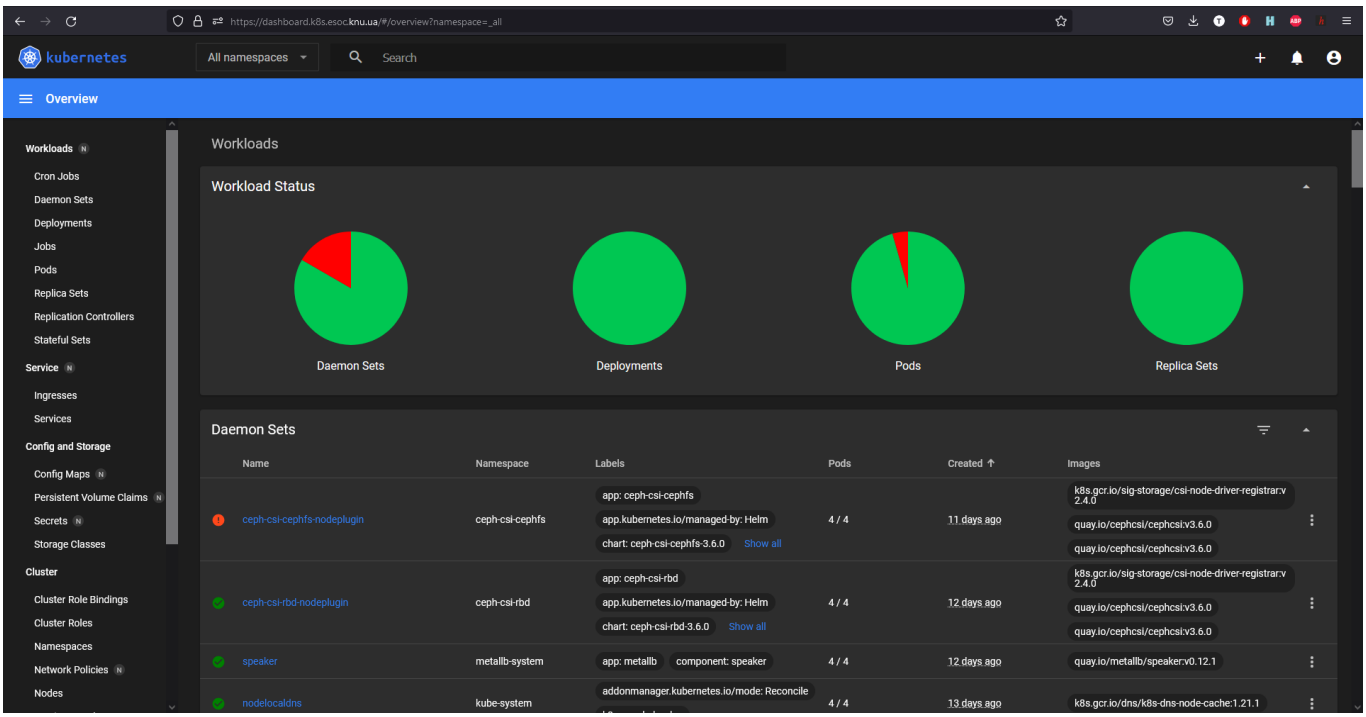

Рисунок 16. Панель керування Kubernetes Dashboard

Найскладнішими етапом є інтеграція Kubernetes кластеру з Ceph кластером для забезпечення постійного сховища.

В даному випадку буде застосовуватись тип сховища в кластері CephFS. Основною перевагою є можливість доступу до даних одночасно з різних сервісів і навіть віртуальних машин. Це надасть можливість робити резервні копії даних і відновити їх у випадку втрати.

Як нотатка для системного адміністратора була створенна дана стаття: https://gl.icc.knu.ua/esoc/kubernetes/ceph-backup.

Для інсталяції потрібно виконати Helm chart з заданими змінними (https://gl.icc.knu.ua/esoc/kubernetes/ceph-backup/-/blob/main/ceph-fs/cephfs.yml). На даному етапі будуть створені провізіонери і сервіси моніторнигу, які будуть відповідати за сторення і доступ до постійного сховища.

Далі потрібно створити Secret. Це маніфест,який створює ключ для доступу Kubernetes кластера до Ceph кластера. Приклад даного маніфесту наведений за посиланням https://gl.icc.knu.ua/esoc/kubernetes/ceph-backup/-/blob/main/ceph-fs/secret.yaml.

І в кінці потрібно описати клас сховища, створивши відповідний маніфест. Також наведений за посиланням https://gl.icc.knu.ua/esoc/kubernetes/ceph-backup/-/blob/main/ceph-fs/storageclass.yaml. Тепер є можливість створювати постійні сховища, вказавши клас сховища як csi-cephfs-sc.

Отже, на даному етапі наш кластер і всі супутні сервіси готові до розгортання ядра всієї системи, а саме сервісу S3 Minio.

3.3. Розгортання сховища бінарних об'єктів

Сервіси сховища бінарних об’єктів будуть розгорнуті в системі оркестраіції контейнерів Kubernetes. Дані зберігатимуться на розподіленій файловій системі Ceph.

Так як всі сервіси будуть знаходитись у внутрішній мережі університету, то потрібно налаштувати доступ до сервісі сховища з мережі інтернет. Для цього потрібно налаштувати Reverse Proxy.

3.3.1. Розгортання Minio

Перш за все потрібно ініціалізувати сховище для збереження даних. Це буде постійне сховище на 45 терабайт. Для цього потрібно застосувати наступний маніфест, який також і створить неймспейс для розгортання сервісу:

https://gl.icc.knu.ua/esoc/kubernetes/minio/-/blob/main/pvc.yaml

```
EL7 VM KNU ESOC K8S root@node1:[~/dashboard]# kubectl get pvc -n minio
NAME              STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS    AGE
minio-cephfs-pvc  Bound    pvc-c67201a0-b91f-4530-a73f-b02972812dd9   45Ti       RWX            csi-cephfs-sc   13d
EL7 VM KNU ESOC K8S root@node1:[~/dashboard]#
```

Рисунок 17. Виведення створеного постійного сховища

Розгортання даного сервісу відбувається за допомогою Helm chart. Для початку потрібно описати всі зміні. Приклад всіх змін, які потрібно внести у файл зі змінними наведені за посиланням https://gl.icc.knu.ua/esoc/kubernetes/minio/-/blob/main/values.yml.

Далі потрібно запусти виконаная Helm chart.

```
helm install --namespace minio --values values.yml --generate-name minio/minio
```

Результатом виконання буде створення наступних сервісів і повна роботоспроможність сервісу Minio:

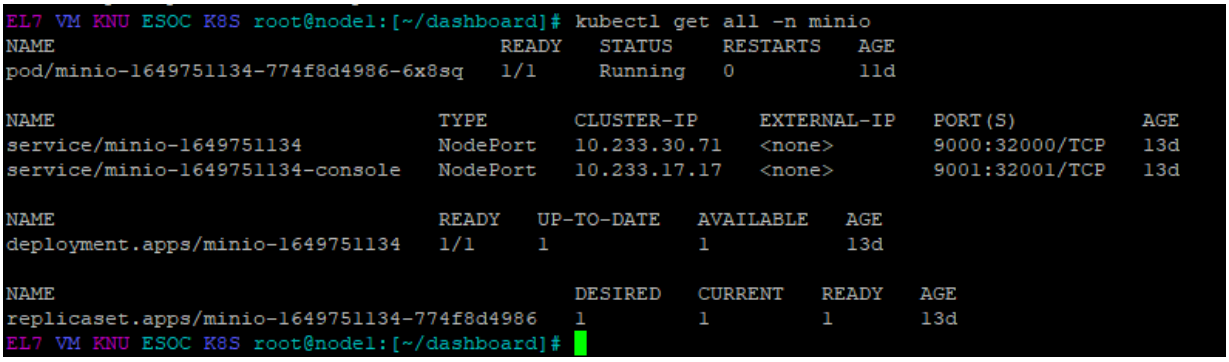

Рисунок 18. Виведення сервісів minio у Kubernetes

3.3.2. Налаштування ReverseProxy

Отже, на даному етапі ми маємо працюючий сервіс minio, але який працює тільки з внутрішньої мережі університету. Потрібно налаштувати проксування запитів на Kubernetes кластер.

В інфраструктурі університету є хости, які мають «білу» адресу, тобто до них є доступ з зовнішньої мережі, мережі інтернет. Через один такий хост буде проксуватись всі запити на сервіси.

Отже будуть використовуватись 3 адреси: s3.k8s.esoc.knu.ua (для доступу до сервісу S3), minio.k8s.esoc.knu.ua (для доступу до панелі керування) та dashboard.k8s.esoc.knu.ua (для доступу до панелі Kubernetes).

Для перенаправлення запитів використовується балансувальник HaProxy. Для цього потрібно додати такі записи до конфігураційного файлу Haproxy:[14]

```
use_backend s3.k8s if { hdr(host) -i
s3.k8s.esoc.knu.ua s3.k8s.esoc.knu.ua:443 }
use_backend console.s3.k8s if { hdr(host) -i
minio.k8s.esoc.knu.ua minio.k8s.esoc.knu.ua:443 }
use_backend k8s-dashboard if { hdr(host) -i
dashboard.k8s.esoc.knu.ua } { src 10.0.0.0/8}

backend s3.k8s
    http-check expect rstatus 200|401
```

```
    option     httpchk     GET     /     HTTP/1.1\r\nHost:\
s3.k8s.esoc.knu.ua
    server  s3-k8s  10.10.109.85:32000 #send-proxy-v2 #check
inter 60000 fastinter 100 downinter 20000
    #server   s3-k8s   10.10.109.186:32000   #send-proxy-v2
#check inter 60000 fastinter 100 downinter 20000
    #server  s3-k8s  10.10.109.140:80 #send-proxy-v2  #check
inter 60000 fastinter 100 downinter 20000
backend  console.s3.k8s
    http-check expect rstatus 200|401
    option     httpchk     GET     /     HTTP/1.1\r\nHost:\
minio.k8s.esoc.knu.ua
    server k8s-s3-console 10.10.109.85:32001 #send-proxy-v2
#check inter 60000 fastinter 100 downinter 20000
    #server  k8s-s3-console  10.10.109.140:80  #send-proxy-v2
#check inter 60000 fastinter 100 downinter 20000

backend k8s-dashboard
    http-check expect rstatus 200|401
    option     httpchk     GET     /     HTTP/1.1\r\nHost:\
dasboard.k8s.esoc.knu.ua
    server k8s-mgr 10.10.109.85:30011 ssl verify none check
inter 60000 fastinter 100 downinter 20000
    server k8s-mgr 10.10.109.86:30011 ssl verify none check
inter 60000 fastinter 100 downinter 20000
    server k8s-mgr 10.10.109.87:30011 ssl verify none check
inter 60000 fastinter 100 downinter 20000
    server k8s-mgr 10.10.109.88:30011 ssl verify none check
inter 60000 fastinter 100 downinter 20000
```

Після цього є можливість отримувати доступ до потрібних сервісів і тестувати та використовувати їх.

3.3.3. Тестування комплексної системи зберігання бінарних об’єктів

На даному етапі потрібно протестувати роботу нашого сервісу.

Спробуємо завантажити декілька файлів а потім отримати доступ до цих файлів.

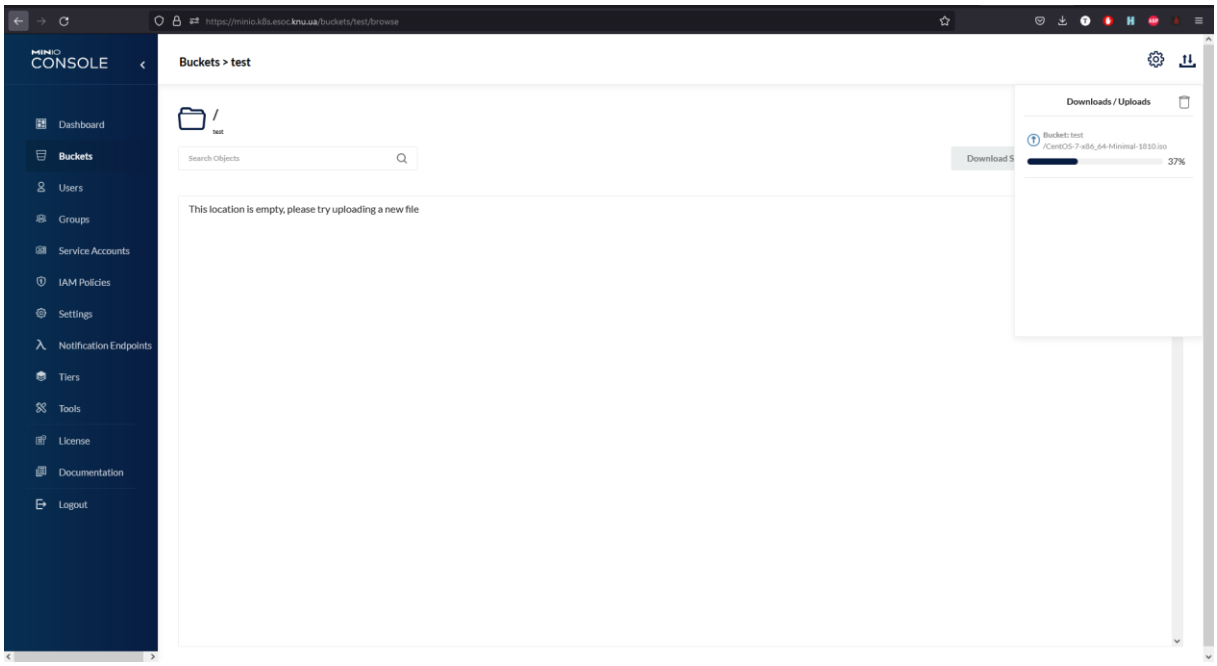

Рисунок 19.Приклад завантаження файла у бакет Minio

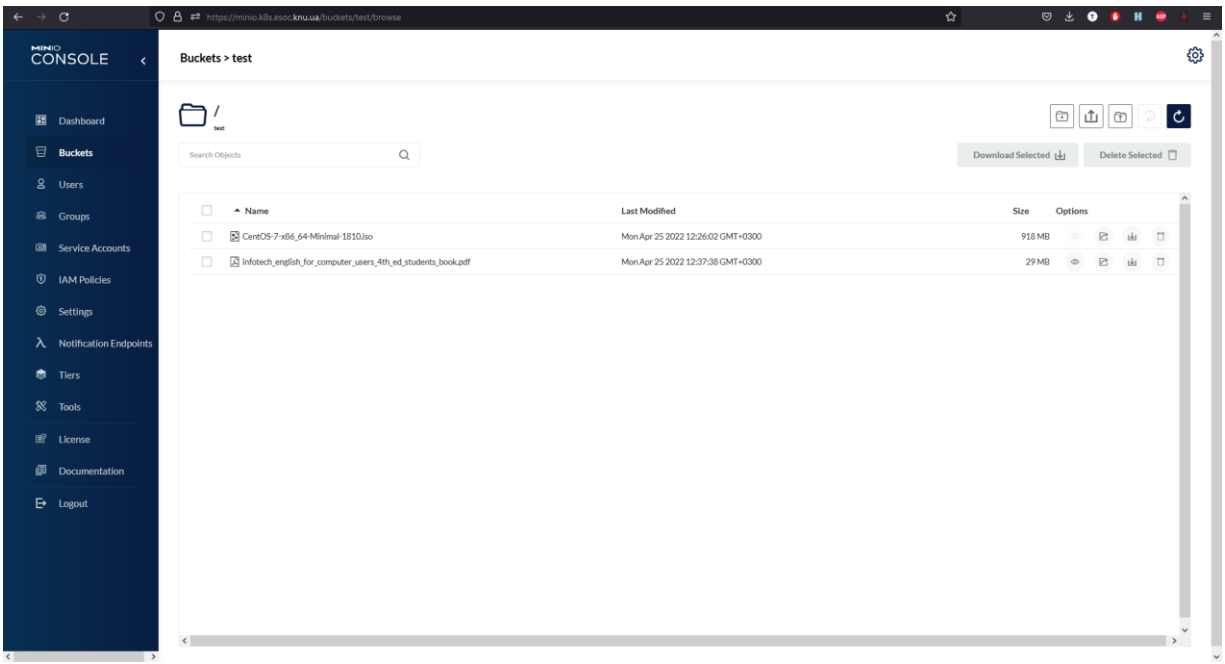

Рисунок 20. Виведення файлів у бакеті Minio

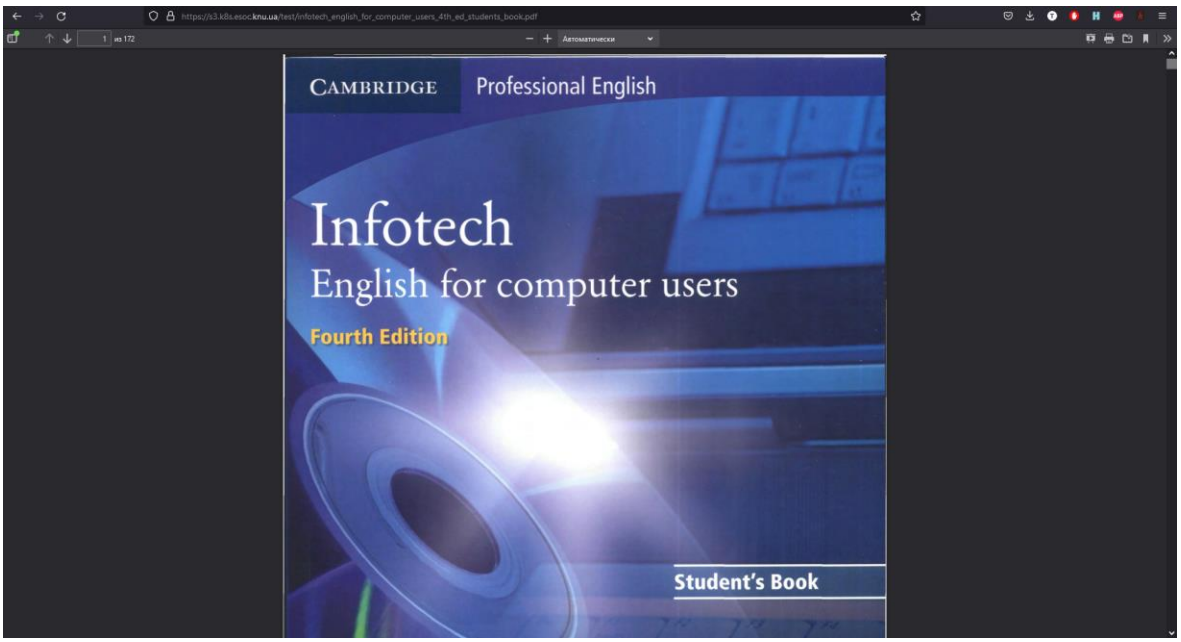

Рисунок 21. Доступ до файла за посиланням Minio

На останньому скріншоті видно, що дуже просто отримати доступ до файла за посиланням. Це означає, що можна розміщувати такі pdf файли на бакеті, і потім інтегрувати їх з веб сторінками або застосунками.

Далі, спробуємо зімітувати помилку в додатку, яке призведе до завершення цього додатку. Для цього ми будемо використовувати Kubernetes Dashboard. Відкриємо потрібний неймспейс і видалимо pod з сервісом minio.

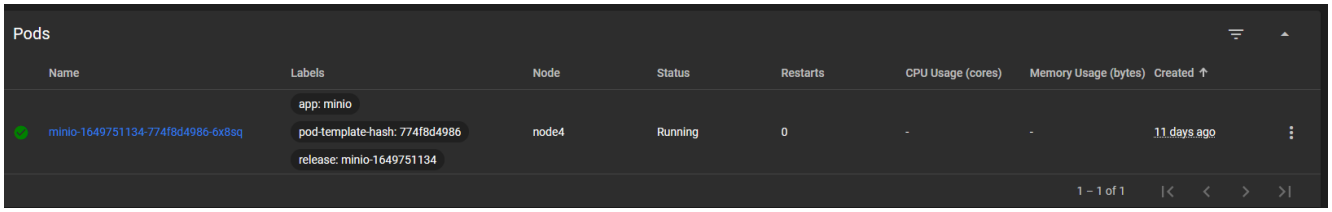

Рисунок 22. Виведення статусу pod Minio у Kubernetes Dashboard

Звернемо увагу, що pod був створений 11 діб тому. Завершимо його роботу.

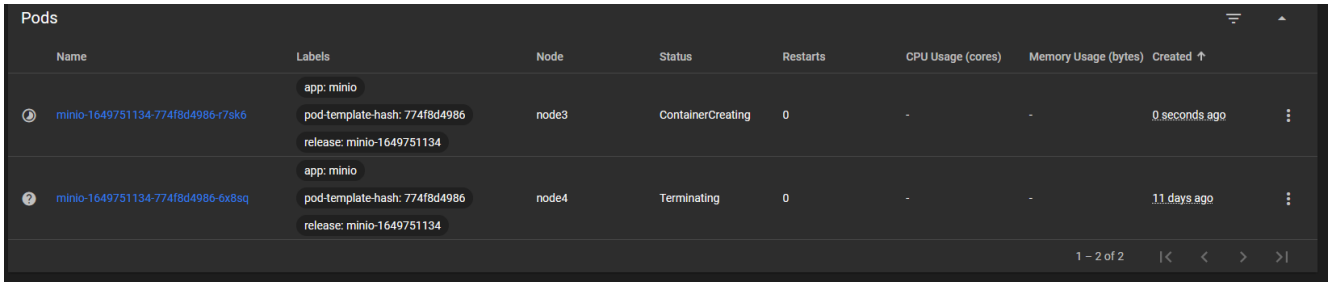

Рисунок 23. Виведення статусу завершення pod Minio у Kubernetes Dashboard

Одразу бачимо почав створюватись новий pod.

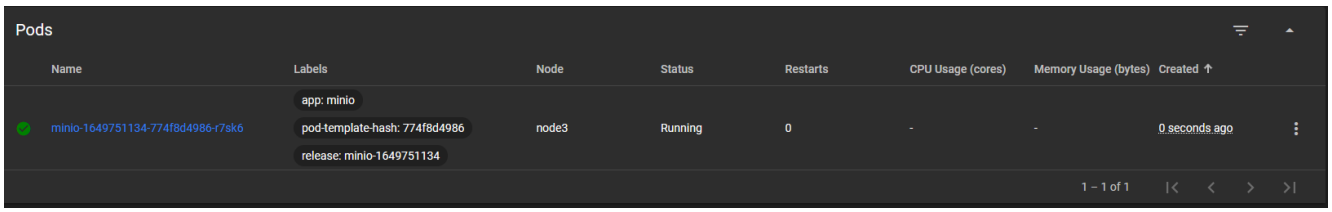

Рисунок 24. Новий pod Minio

Отже, при аварійному завершенні застосунку, він буде автоматично перезапущений.

Далі спробуємо відключити вузел кластера, на якій запущений цей контейнер. Це 3 вузол кластеру.

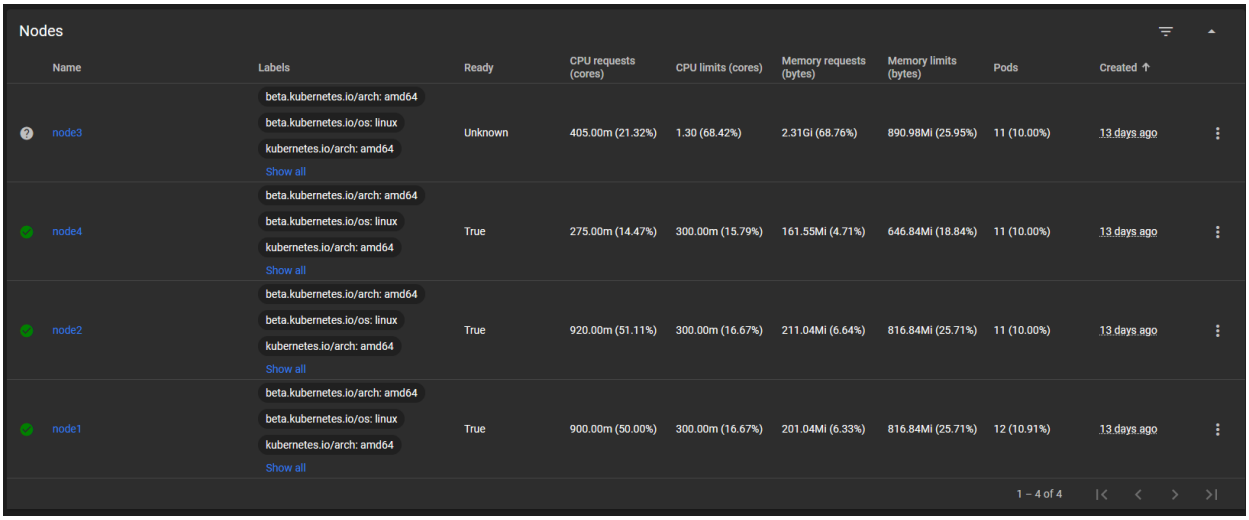

Рисунок 25. Завершення роботи вузла кластера

При виключенні вузла кластера нам сигналізують відповідні позначки на панелі, а також було отримане відповідне сповіщення від служби моніторингу.

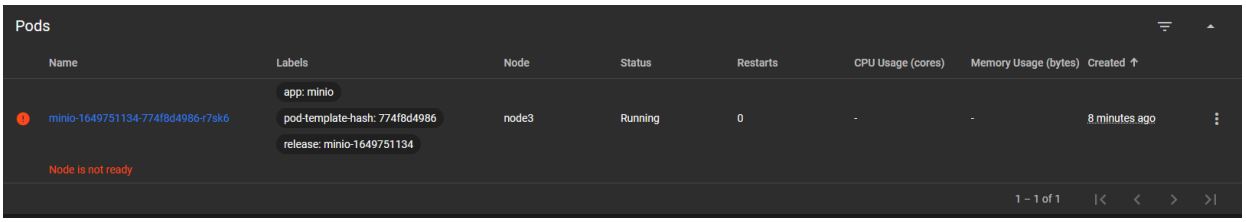

Рисунок 26. Недоступність Pod Minio

Pod був деякий час недоступний, після чого був автоматично перезапущений службами Kubernetes.

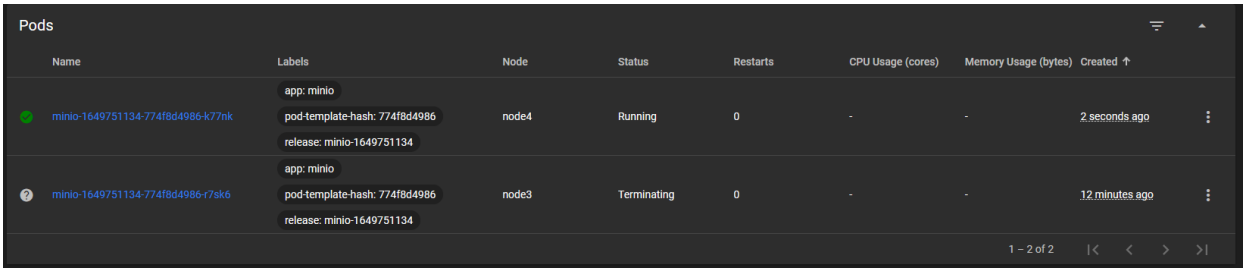

Рисунок 27. Перестворення сервісу на новому вузлі кластера

Отже, можна сказати, що ця система сховища бінарних об’ктів є достатньо надійною і не потребує великих втручань адміністратора у випадку збою.

Загалом було проведене комплексне тестування і результати цього тестування наведені в таблиці 3.3.3.1.

Таблиця 3.3.3.1

Непередбачувана ситуація	Можливість читання файлів	Можливість запису файлів	Можливість створення корзин для файлів	Кількість можливих з’єднанань
Збій процесу minio.	Не впливає. Майже одразу є доступ до читання файлів.	Не впливає. Майже одразу є доступ до запису файлів.	Не впливає. Майже одразу є можливість створювати корзини.	Не впливає. Більше 10 000 з’єднань.
Зависання одного з виконавчих вузлів кластеру	Читання файлів буде недоступне протягом декількох хвилин після збою, потім автоматично відновиться	Запис файлів буде недоступним протягом декількох хвилин після збою, потім автоматично відновиться	Створення корзин для файлів буде недоступним протягом декількох хвилин після збою, потім автоматично відновиться	З’єднання будуть відкидатись протягом декількох хвилин, потім особливого впливу не помітимо.

Зависання керуючого вузла кластеру.	Не впливає на доступ до читання файлів.	Не впливає на доступ до запису файлів.	Не впливає на можливість створювати нові корзини для файлів	Не впливає. Більше 10 000 з’єднань.
Високе навантаження на вузол кластера	Не впливає на доступ до читання файлів.	Не впливає на доступ до запису файлів	Не впливає на можливість створювати нові корзини для файлів	Кількість можливих одночасних з’єднань спаде.
Вихід з ладу одного з дисків	Не впливає на доступ до читання файлів.	Не впливає на доступ до запису файлів	Не впливає на можливість створювати нові корзини для файлів	Не впливає. Більше 10 000 з’єднань.
Вимкнення електрики в серврній	Відсутній доступ	Відсутній доступ	Відсутній доступ	З’єднання не можливі

ВИСНОВКИ

Аналіз засобів зберігання даних та типових програмних комплексів у вищих навчальних закладах показав доцільність використання систем зберігання бінарних об’єктів даних.

Запропонована архітектура дозволяє забезпечити стандартний поширений тип взаємодії з програмними комплексами, необхідний рівень доступності та швидкодії.

Апробація розгорнутої системи дозволила підтвердити відповідність запропонованого рішення вимогам сховища бінарних об’єктів в IT інфраструктурі університету.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bill Karwin. SQL Antipatterns. Avoiding the Pitfalls of Database Programming.
2. Andrew S. Tanenbaum, Maarten van Steen. DISTRIBUTED SYSTEMS PRINCIPLES AND PARADIGMS.
3. J.T. Wolohan. Object Storage Across the Cloud AWS, Azure and GCP.
4. А. Ізмайлова. Масштабування баз даних
5. Зберігання бінарних об’єктів на прикладі Minio [Електронний ресурс].- Режим доступу: URL: https://habr.com/company/ozontech/blog/586024/
6. Структурована та неструктурована інформація[Електронний ресурс].- Режим доступу: URL: https://www.purestorage.com/knowledge/big-data/structured-vs-unstructured-data.html
7. Що таке хмарне сховище? [Електронний ресурс].- Режим доступу: URL: https://aws.amazon.com/us/what-is-cloud-storage/
8. Sage A. Weil. CEPH: RELIABLE, SCALABLE, AND HIGH-PERFORMANCE DISTRIBUTED STORAGE
9. Жугастров О.О. Хмарні обчислення: сутність, недоліки, переваги
10. Барабаш О. В., Куліковська Ю. А. Аналіз експлуатаційних проблем розподілених систем.
11. Лукша М. Kubernetes у дії
12. What is the Ceph File System (CephFS)? [Електронний ресурс]. – Режим доступу: URL: https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/2/html/ceph_file_system_guide_technology_preview/what_is_the_ceph_file_system_cephfs.
13. High-Performance Object Storage Solutions. [Електронний ресурс]. – Режим доступу: URL: https://www.seagate.com/gb/en/solutions/partners/minio/
14. HAProxy as a reverse proxy. [Електронний ресурс]. – Режим доступу: URL: https://blog.cloudant.com/2020/10/27/HAProxy-as-a-reverse-proxy.html
15. MinIO High Performance Object Storage. [Електронний ресурс]. – Режим доступу: URL: https://docs.min.io/minio

16. Eliezer Levy, Abraham Silberschatz. Distributed file systems: concepts and examples