СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____

Кафедра _____інформаційних технологій та програмування_____

Пояснювальна записка

до магістерської дипломної роботи

_____магістр_____
(освітньо-кваліфікаційний рівень)

на тему: **Розробка хмарного сервісу збереження інформації**

Виконав: студент 2 курсу, групи ІСТ-22дм_ 126 «Інформаційні системи та технології»___
(шифр і назва спеціальності)
_____ Шубін А.С. _____
(прізвище та ініціали)
Керівник ___ Захожай О.І. _____
(прізвище та ініціали)
Рецензент__Меняйленко О.С. _____
(прізвище та ініціали)

Київ – 2023 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____
Кафедра _____інформаційних технологій та програмування_____
Освітньо-кваліфікаційний рівень __магістр_____
Спеціальність 126 «Інформаційні системи та технології»_____
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., доц. Захожай О.І.
(підпис)
« ___ » _____2023 р.

ЗАВДАННЯ

на магістерську дипломну роботу студенту
_____Шубіну Антону Сергійовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка хмарного сервісу збереження інформації,____ керівник роботи доц., д.т.н. Захожай Олег Ігоревич_____ затверджені наказом вищого навчального закладу від 21.11.2023 року №614/15.15-С
2. Строк подання студентом роботи 6 грудня 2023 р.
3. Вихідні дані до роботи: Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі .
   3.1 Літературні джерела:
   1. Ніколас Карр. The Big Switch: Rewiring the World, from Edison to Google.
   2. Аршдіп Бага, Віджей Медісетті. Cloud Computing: A Hands-On Approach.
   3. Майкл Дж. Кевіс. The Art if Cloud Computing: Advanced Topics in Scalable Web Development.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1 Вступ
   4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:
   Огляд хмарних технологій.
   Інфраструктура хмарного середовища.
   Особливості серверних баз даних.
   4.3 Основна частина, в якій висвітлити: створення і налаштування інфраструктури хмарного середовища, програмну реалізацію

програмного комплексу з можливістю збереження та поширення даних через віддалене сховище даних.

4.4 Висновки

4.5 Перелік використаних джерел

5. Перелік графічного матеріалу немає
6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання 20 жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	20.10.23	
2	Укладання і погодження з керівником плану і етапів виконання роботи	23.10.23	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	30.10.23	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	08.11.23	
5	Проектування інформаційної моделі задачі, що реалізується	18.11.23	
6	Укладання та тестування програмного продукту	28.11.23	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	05.12.23	
8	Здача готової пояснювальної записки на кафедру	06.12.23	
9	Укладання доповіді і презентації	12.12.23	

Студент _____ Шубін А.С.___
(підпис) (прізвище та ініціали)

Керівник роботи _____ _Захожай О.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текст – 50, рис. – 21, табл. – 0, додатків – 3, літературних джерел – 7

У ході виконання даної дипломної роботи були проведені дослідження предметної області баз даних, операційних систем та програмного забезпечення.

Метою роботи є налаштування інфраструктури та перетворення локального сховища даних у віддалене. Використовуючи операційну систему, веб-сервер, серверну базу даних, систему об’єктного сховища та графічний інтерфейс користувача.

Результати роботи, зокрема, розроблене програмне забезпечення може бути практично використане в багатьох галузях, пов’язаних із забезпеченням користувачів хмарним сховищем даних, даючи можливість зберігати дані та мати до них доступ без навантаження на локальні ресурси, такі як жорсткі диски або флеш-пам’ять.

Ключові слова: ХМАРНІ ТЕХНОЛОГІЇ. ВІДДАЛЕНЕ СХОВИЩЕ ДАНИХ. СЕРВЕР. БАЗА ДАНИХ.

ЗМІСТ

ВСТУП

Актуальність досліджень. В наш сучасний час кожна людина володіє пристроєм, що здатен зберігати інформацію. Комп'ютер, смартфон, фото- або відео-камера – все це пов'язано можливостю створення, збереження, та поширення інформації. З роками експлуатації ці пристрої можуть виходити з ладу, та деяка частка даних користувача втрачається, в деяких випадках - без можливості їх відновлення. Документи, фото, відео, аудіо – щоб зберегти всі ці дані, користувачеві пропонується використовувати хмарне середовище для збереження інформації. Існують такі популярні рішення, як Google Drive, Dropbox, Microsoft OneDrive або Amazon S3. Існує лише одна проблема – користувач може не погодитися з політикою конфіденційності цих сервісів, поставить під сумнів безпекову цілісність середи, тощо.

Об'єкт досліджень: хмарне середовище зберігання даних.

Предмет досліджень: інфраструктура хмарного середовища

Мета дослідження: використання операційної системи, бази даних та веб-сервера, створення графічного інтерфейсу користувача, що забезпечить віддалений доступ до даних.

Завдання дослідження:

a) провести аналітичний огляд літератури по серверних операційних системах;

b) виділити переваги і недоліки сумісних серверних баз даних;

c) налаштувати інфраструктуру, що складається з веб-серверу, серверної бази даних, об'єктної середи сховища;

d) створення графічного інтерфейсу користувача для полегшення взаємодії зі збереженими даними.

Методологічна й теоретична основа дослідження: хмарні технології та бази даних.

Практичне значення отриманих результатів. Отримані результати роботи можуть бути використані в різних галузях, пов’язаних із забезпеченням користувачів дистанційним доступом до збереженої даних, а розроблене програмне забезпечення дозволить використовувати будь-який локальний ресурс у якості хмарного сховища даних.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ РОЗРОБКИ ХМАРНОГО СЕРЕДОВИЩА

1.1 Огляд інфраструктури хмарного середовища

Хмарне сховище – це модель збереження даних у комп’ютері, в якій цифрові дані накопичуються в логічні пули, а фізичне зберігання охоплює кілька серверів, зазвичай у кількох місцях. Фізичне середовище, як правило, належить хостинговим компаніям, які ним керують.

Хмарне сховище базується на високовіртуалізованій інфраструктурі й схоже на ширші хмарні обчислення з точки зору доступних інтерфейсів, майже миттєвої еластичності та масштабованості, багатожитлового використання та дозованих ресурсів. Хмарні сервіси зберігання даних можуть бути використані за межами локальної служби, або розгорнутись у локальних приміщеннях. Хмарне сховище може бути використане за межами підприємства або на самому підприємстві. Хмарне сховище, як правило, належить до хостингових сервісів зберігання об’єктів.

Платформи для зберігання, такі як Amazon S3 і Microsoft Azure Storage, програмне забезпечення таке, як OpenStack Swift, системи зберігання об’єктів такі, як EMC Atmos, EMC ECS і Hitachi Content Platform – усі приклади зберігання, що відповідають характеристикам хмарних систем схову даних.

Хмарне сховище:

- зроблене з багатьох розподілених ресурсів, але виступає як одне ціле – часто називають хмарним зберіганням або федеративним;
- висока відмовостійкість унаслідок резервування й розподілу даних;
- висока міцність шліхом створення версій, копій.

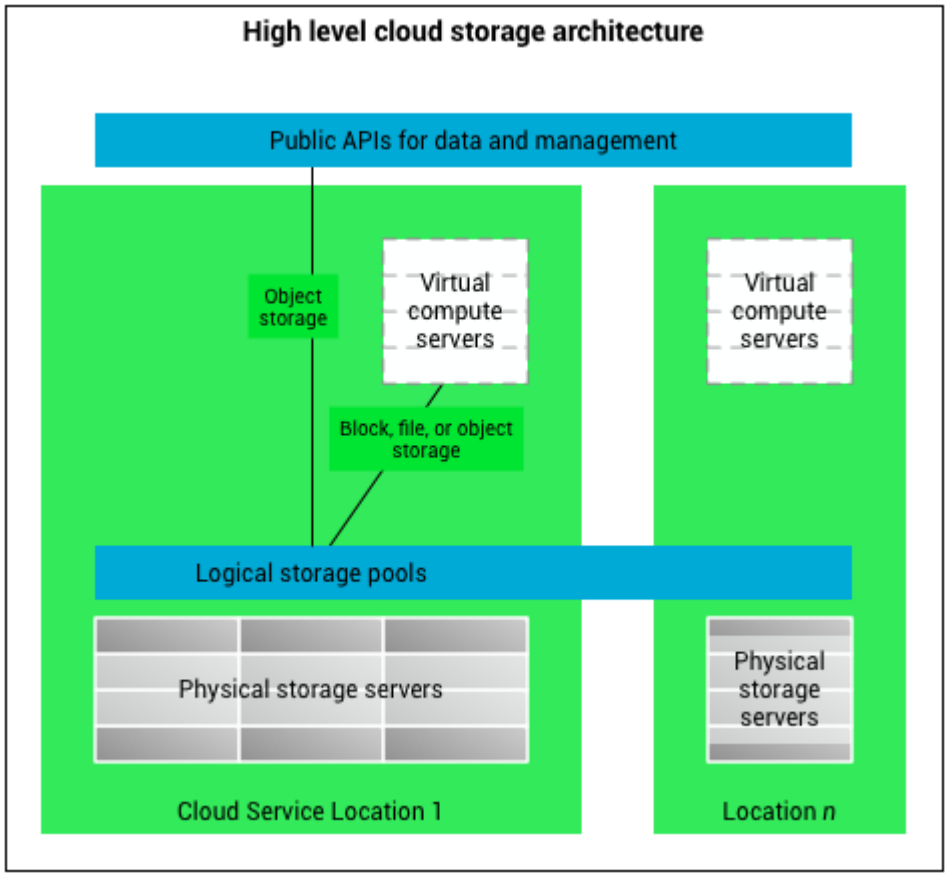


Рис. 1.1. Високорівнева архітектура хмарного сховища

1.2 Гіганти хмарного хостингу

Google Drive – сховище даних, яке належить компанії Google Inc. Головною особливістю є узагальнений офісний пакет, що включає в собі Google документи, Таблиці та Презентації.

Google Drive включає в себе систему обміну файлами, де творець файлу або теки за замовчуванням є його власником. Власник має можливість регулювати видимість файлу або теки для інших користувачів. Право власності підлягає передачі. Файли і теки можуть спільно використовуватись приватно конкретними користувачами, що мають обліковий запис Google, використовуючи свої *@gmail.com* адреси електронної пошти. Результати пошуку можуть бути звужені за типом файлу, власником, видимістю і додатком для перегляду. Google Drive підтримує логічні оператори.

Amazon S3 – сервіс-сховище даних, один з Amazon Web Services. Сервіс надає можливість для зберігання й отримання будь-якого обсягу даних, у будь-який час з будь-якої точки мережі, тобто так званий файловий хостинг. За допомогою Amazon S3 досягається висока масштабність, надійність, висока швидкість, недорога інфраструктура зберігання даних.

Amazon S3 оперує даними через архітектуру об’єктного сховища, що запроваджує доступність, низьку затримку та високу надійність. Базові одиниці сховища Amazon S3 це об’єкти, які організовуються у блоки. Кожний об’єкт ідентифікується унікальним, зазначеним користувачем ключем. Подальша робота з блоками забезпечується консоллю від Amazon S3, програмно через AWS SDK, або з інтерфейсом програмування додатків REST.

Microsoft OneDrive (раніше SkyDrive, Windows Live SkyDrive і Windows Live Folders) – це файловий хостинг, що надається компанією Microsoft як частина набору онлайн-послуг. Він дозволяє користувачам зберігати файли, а також інші особисті дані, такі як налаштування Windows або ключі відновлення BitLocker у хмарі. Файли можна синхронізувати з ПК та отримувати доступ до них з веббраузера або мобільного пристрою, а також ділитися публічно або з певними людьми.
Існує підтримка Office Online в OneDrive. Це дозволяє користувачам завантажувати, створювати, редагувати та обмінюватися документами Microsoft Office безпосередньо в веббраузері. Користувачі можуть створювати, переглядати та редагувати документи Word, Excel, PowerPoint і OneNote прямо в браузері. Безсумнівною перевагою сервісу є можливість запису файлів шляхом простого переміщення або використання вебдодатків. Присутній і віддалений доступ до комп'ютера, який працює під управлінням Windows.

1.3 Серверні операційні системи

Операційна система, скорочено ОС (англ. operating system, OS) — це базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем.

Операційна система звичайно складається з ядра операційної системи та базового набору прикладних програм.

Головні функції:

- Виконання на вимогу користувача тих елементарних (низькорівневих) дій, які є спільними для більшості програм і часто зустрічаються майже в усіх програмах (введення та виведення

даних, запуск і зупинка інших програм, виділення та вивільнення додаткової пам'яті тощо).

- Стандартизований доступ до периферійних пристроїв (пристрої введення-виведення).
- Завантаження програм в оперативну пам'ять і їх виконання.
- Керування оперативною пам'яттю (розподіл між процесами, організація віртуальної пам'яті).
- Керування доступом до даних енергонезалежних носіїв (жорсткий диск, оптичні диски тощо), організованим у тій чи іншій файловій системі.
- Відтворення інтерфейсу користувача.
- Мережеві операції, підтримка стеку мережевих протоколів.

Додаткові функції:

- Паралельне або псевдопаралельне виконання задач (багатозадачність).
- Розподіл ресурсів обчислювальної системи між процесами.
- Організація надійних обчислень (неможливості впливу процесу на перебіг інших), основана на розмежуванні доступу до ресурсів.
- Взаємодія між процесами: обмін даними, синхронізація.
- Захист самої системи, а також даних користувача і програм від дій користувача або інших програм.
- Багатокористувацький режим роботи та розподілення прав доступу (автентифікація, авторизація).

До складу операційної системи входять:

- ядро операційної системи, що забезпечує розподіл та керування ресурсами обчислювальної системи;
- базовий набір прикладних програм, системні бібліотеки та програми обслуговування.

Ядро системи — це набір функцій, структур даних та окремих програмних модулів, які завантажуються в пам'ять комп'ютера

при завантаженні операційної системи та забезпечують три типи системних сервісів:

- керування введенням-виведенням інформації (підсистема вводу-виводу ядра ОС);
- керування оперативною пам'яттю (підсистема керування оперативною пам'яттю ядра ОС);
- керування процесами (підсистема керування процесами ядра ОС).

Кожна з цих підсистем представлена відповідними функціями ядра системи.

Багатозадачні операційні системи також включають ще одну обов'язкову складову — механізм підтримки багатозадачності. Ця складова не надається як системний сервіс і тому не може бути віднесена до жодної з підсистем.

Безпека ОС базується на двох ідеях:

1. ОС надає прямий чи непрямий доступ до ресурсів на кшталт файлів на локальному диску, привілейованих системних викликів, особистої інформації про користувачів та служб, представлених запущеними програмами;
2. ОС може розділити запити ресурсів від авторизованих користувачів, дозволивши доступ, та неавторизованих, заборонивши його.

Запити, в свою чергу, також діляться на два типи:

1. Внутрішня безпека — вже запущені програми. На деяких системах програма, оскільки вона вже запущена, не має ніяких обмежень, але все ж типово вона має ідентифікатор, котрий використовується для перевірки запитів до ресурсів.
2. Зовнішня безпека — нові запити з-за меж комп'ютера, як наприклад реєстрація з консолі чи мережеве з'єднання. В цьому випадку відбувається процес авторизації за допомогою імені

користувача та паролю, що його підтверджує, чи інших способів як наприклад магнітні картки чи біометричні дані.

На додачу до моделі дозволити/заборонити системи з підвищеним рівнем безпеки також слідкують за діяльністю користувачів, що дозволяє пізніше дати відповідь на питання типу «Хто читав цей файл?»

1.3.1 Ubuntu Server OS

Операційна система на базі Linux, що складається переважно з безкоштовного програмного забезпечення з відкритим кодом. Ubuntu офіційно випущено в кількох версіях: Desktop, Server, Core для Internet of Things пристроїв і роботів.

Як і в інших дистрибутивах Linux, усі випуски можна запускати окремо на комп'ютері або у віртуальній машині (наприклад, WSL у Microsoft Windows). Оновлення до Ubuntu випускається кожні шість місяців, а довгострокова підтримка (LTS) — кожні два роки. Canonical надає оновлення безпеки та підтримку для кожного випуску Ubuntu, починаючи з дати випуску й доки не досягне визначеної дати завершення життєвого циклу (EOL). Canonical отримує дохід за рахунок продажу послуг преміум-класу, пов'язаних з Ubuntu, і пожертвувань тих, хто завантажує програмне забезпечення Ubuntu.

Стандартна інсталяція Ubuntu від версії 23.10 містить мінімальний вибір програмного забезпечення, а саме веб-браузер (Firefox) і основні утиліти GNOME. Багато додаткових програмних пакетів доступні за допомогою вбудованого програмного забезпечення Ubuntu (раніше Ubuntu Software Center), а також будь-яких інших інструментів керування пакетами на основі APT. Багато додаткових програмних пакетів, які більше не встановлені за замовчуванням, наприклад Evolution, GIMP, Pidgin і Synaptic, все ще доступні в репозиторіях і можуть бути встановлені за допомогою основного інструменту або будь-якого іншого інструменту

керування пакетами на основі APT. Також доступні крос-дистрибутивні пакети знімків і Flatpaks, які дозволяють інсталювати програмне забезпечення, як-от деяке програмне забезпечення Microsoft, у більшості основних операційних систем Linux. Файловим менеджером за замовчуванням є GNOME Files, який раніше називався Nautilus.

Ubuntu прагне бути безпечним за замовчуванням. Програми користувача працюють із низькими привілеями та не можуть пошкодити операційну систему чи файли інших користувачів. Для підвищення безпеки інструмент sudo використовується для призначення тимчасових привілеїв для виконання адміністративних завдань, що дозволяє обліковому запису root залишатися заблокованим і допомагає недосвідченим користувачам випадково внести катастрофічні зміни в систему або відкрити діри в безпеці. Polkit також широко впроваджується в робочий стіл.

Більшість мережевих портів закриті за замовчуванням, щоб запобігти злому. Вбудований брандмауер дозволяє кінцевим користувачам, які встановлюють мережеві сервери, контролювати доступ. Для його налаштування доступний графічний інтерфейс користувача (GUI for Uncomplicated Firewall). Ubuntu компілює свої пакунки, використовуючи такі функції GCC, як PIE і захист від переповнення буфера, щоб посилити програмне забезпечення. Ці додаткові функції значно підвищують безпеку за рахунок зниження продуктивності на 0,01% у 64-бітній версії.

Ubuntu також підтримує повне шифрування диска, а також шифрування домашніх і приватних каталогів.

1.4 Сервера баз даних

Сервер бази даних — це сервер, який використовує програму бази даних, яка надає послуги бази даних іншим комп’ютерним програмам або комп’ютерам, як визначено моделлю клієнт-сервер. Системи керування базами даних (СУБД) часто надають функції сервера баз даних, а деякі

системи керування базами даних (такі як MySQL) покладаються виключно на модель клієнт-сервер для доступу до бази даних (у той час як інші, наприклад SQLite, призначені для використання як вбудована база даних).

Користувачі отримують доступ до сервера бази даних або через «фронтенд», який працює на комп'ютері користувача, який відображає запитувані дані, або через «бек-енд», який працює на сервері та виконує такі завдання, як аналіз і зберігання даних.

У моделі «головний-підлеглий» головні сервери баз даних є центральним і основним розташуванням даних, тоді як підлеглі сервери баз даних є синхронізованими резервними копіями головного, що діє як проксі.

Більшість програм баз даних відповідають на мову запитів. Кожна база даних розуміє свою мову запитів і перетворює кожен надісланий запит у форму, зрозумілу серверу, і виконує його для отримання результатів.

Приклади власних програм баз даних включають Oracle, IBM Db2, Informix і Microsoft SQL Server. Приклади безкоштовних програм баз даних включають PostgreSQL; і під GNU General Public License включають Ingres і MySQL. Кожен сервер використовує власну логіку та структуру запитів. Мова запитів SQL (Structured Query Language) більш-менш однакова для всіх програм реляційної бази даних.

Для роз'яснення, сервер бази даних — це просто сервер, який підтримує служби, пов'язані з клієнтами, через програми баз даних.

1.4.1 MySQL Database Server

Вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все

розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосунків. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- Простота у встановленні та використанні;
- Підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- Кількість рядків у таблицях може сягати 50 млн;
- Висока швидкість виконання команд;
- Наявність простої і ефективної системи безпеки.

1.5 Веб-сервер

Веб-сервер — це комп’ютерне програмне забезпечення та базове обладнання, яке приймає запити через HTTP (мережевий протокол, створений для розповсюдження веб-вмісту) або його безпечний варіант HTTPS. Агент користувача, як правило, веб-браузер або веб-сканер, ініціює зв’язок, надаючи запит на веб-сторінку чи інший ресурс за допомогою HTTP, і сервер відповідає вмістом цього ресурсу або повідомленням про помилку. Веб-сервер також може приймати та зберігати ресурси, надіслані від агента користувача, якщо на це налаштовано.

Апаратне забезпечення, яке використовується для роботи веб-сервера, може відрізнятися залежно від обсягу запитів, які він повинен обробляти. У нижній частині діапазону знаходяться вбудовані системи, такі як маршрутизатор, який запускає невеликий веб-сервер як інтерфейс конфігурації. Веб-сайт із високим трафіком може обробляти запити з сотень серверів, які працюють на стійках із високошвидкісними комп'ютерами.

Ресурс, надісланий з веб-сервера, може бути вже існуючим файлом (статичний вміст), доступним для веб-сервера, або він може бути згенерований під час запиту (динамічний вміст) іншою програмою, яка спілкується з програмним забезпеченням сервера. Перші зазвичай можуть обслуговуватися швидше та легше зберігатися в кеш-пам'яті для повторних запитів, тоді як другі підтримують ширший діапазон програм.

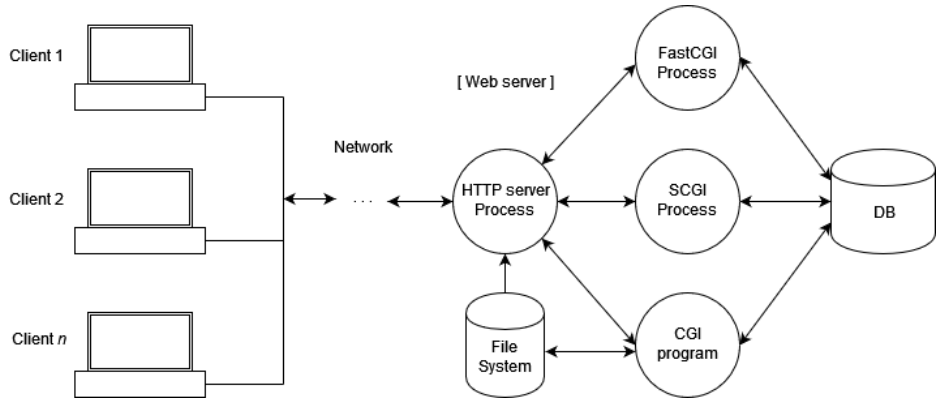


Рис. 1.2 Клієнти ПК, які спілкуються через мережу з веб-сервером, який обслуговує статичний і динамічний вміст

Такі технології, як REST і SOAP, які використовують HTTP як основу для загального зв'язку між комп'ютерами, а також підтримують розширення WebDAV, розширили застосування веб-серверів далеко за межі їх первісної мети – обслуговування сторінок, які читаються людиною.

1.5.1 Nginx Web Server

Nginx — це веб-сервер, який також можна використовувати як зворотний проксі, балансувальник навантаження, поштовий проксі та HTTP-кеш. Nginx — це безкоштовне програмне забезпечення з відкритим вихідним кодом, випущене згідно з умовами ліцензії BSD із двох пунктів. Велика частина веб-серверів використовує Nginx, часто як балансувальник навантаження.

Nginx легко налаштувати, щоб обслуговувати статичний веб-вміст або діяти як проксі-сервер.

Nginx можна розгорнути, щоб також обслуговувати динамічний вміст у мережі за допомогою FastCGI, обробників SCGI для сценаріїв, серверів додатків WSGI або модулів Phusion Passenger, і може служити програмним балансувальником навантаження.

Nginx використовує асинхронний підхід, керований подіями, а не потоки для обробки запитів. Модульна керована подіями архітектура Nginx може забезпечити передбачувану продуктивність за високих навантажень.

Nginx був написаний з явною метою перевершити веб-сервер Apache. Хоча в минулому Nginx перевершував Apache, починаючи з Apache 2.4, вони пропонують подібну продуктивність. Цей попередній приріст продуктивності відбувся ціною зниження гнучкості, наприклад, можливості перевизначати загальносистемні налаштування доступу для кожного окремого файлу (Apache досягає цього за допомогою файлу .htaccess, тоді як Nginx не має такої вбудованої функції).

Раніше додавання сторонніх модулів до Nginx вимагало перекомпіляції програми з вихідного коду зі статичним зв’язуванням модулів. Це було частково усунено у версії 1.9.11 у лютому 2016 року з додаванням динамічного завантаження модуля. Однак модулі все одно повинні бути скомпільовані одночасно з Nginx, і не всі модулі сумісні з цією системою; деякі вимагають старішого процесу статичного зв’язування.

1.6 Сховища об’єктів

Об’єктне сховище (також відоме як об’єктно-орієнтоване сховище або сховище blob) – це комп’ютерний підхід до зберігання даних, який керує даними як «блобами» або «об’єктами», на відміну від інших архітектур зберігання, таких як файлові системи, які керують даними як файловою ієрархією, і блочне сховище, яке керує даними як блоками в секторах і доріжках. Кожен об’єкт зазвичай пов’язаний із змінною кількістю метаданих і глобальним унікальним ідентифікатором.

Зберігання об’єктів може бути реалізовано на кількох рівнях, включаючи рівень пристрою (пристрій зберігання об’єктів), рівень системи та рівень інтерфейсу. У кожному разі об’єктне сховище прагне забезпечити можливості, які не застосовуються іншими архітектурами сховища, як-от інтерфейси, які безпосередньо програмуються програмою, простір імен, який може охоплювати кілька екземплярів фізичного обладнання, і функції керування даними, такі як реплікація даних і розподіл даних у деталізація на рівні об'єкта.

Системи зберігання об’єктів дозволяють зберігати величезні обсяги неструктурованих даних, у яких дані записуються один раз і зчитуються один (або багато разів). Зберігання об’єктів використовується для таких цілей, як зберігання відео та фотографій у Facebook, пісень у Spotify або файлів у онлайн-сервісах для спільної роботи, таких як Dropbox. Одним із обмежень об’єктного сховища є те, що воно не призначене для транзакційних даних, оскільки об’єктне сховище не призначене для заміни доступу до файлів NAS і спільного використання; він не підтримує механізми блокування та спільного використання, необхідні для підтримки єдиної, точно оновленої версії файлу.

Одним із принципів розробки об’єктного сховища є абстрагування деяких нижніх рівнів сховища від адміністраторів і програм. Таким чином, дані представлені та керовані як об’єкти замість блоків або (виключно)

файлів. Об’єкти містять додаткові описові властивості, які можна використовувати для кращого індексування або керування. Адміністраторам не потрібно виконувати функції зберігання нижчого рівня, як-от створення та керування логічними томами для використання ємності диска або налаштування рівнів RAID для вирішення проблем диска.

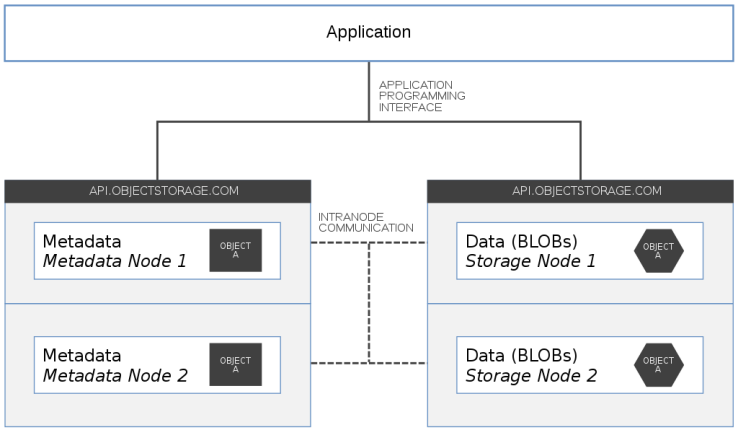


Рис 1.3. Високорівнева архітектура сховища об’єктів

Зберігання об’єктів також дозволяє адресувати та ідентифікувати окремі об’єкти не тільки за назвою файлу та шляхом до нього. Зберігання об’єктів додає унікальний ідентифікатор у сегменті або в усій системі, щоб підтримувати набагато більші простори імен і усунути колізії імен.

1.6.1 MinIO Object Storage

MinIO — це високопродуктивне сховище об’єктів, випущене згідно з GNU Affero General Public License v3.0. Він сумісний з API-службою хмарного сховища Amazon S3. Він здатний працювати з неструктурованими даними, такими як фотографії, відео, файли журналів, резервні копії та зображення контейнерів із максимальним підтримуваним розміром об’єкта 50 ТБ.

Стек зберігання MinIO складається з трьох основних компонентів: MinIO Server, MinIO Client (він же mc, який є клієнтом командного рядка для керування об’єктами та файлами з будь-якими сумісними серверами Amazon

S3) і MinIO Client SDK, який можуть використовувати розробники програм. для взаємодії з будь-яким сумісним сервером Amazon S3.

Хмарний сервер зберігання даних MinIO розроблений як мінімальний і масштабований. Він досить легкий, щоб бути в комплекті зі стеком програм, подібно до NodeJS і Redis.

MinIO оптимізовано для розгортання великих підприємств, включаючи такі функції, як кодування стирання, захист від bitrot, шифрування/WORM, керування ідентифікацією, безперервна реплікація, глобальна федерація та багатохмарне розгортання через режим шлюзу.

Сервер MinIO не залежить від апаратного забезпечення, тому його можна інсталювати як на фізичних, так і на віртуальних машинах або запускати як контейнери Docker і розгортати на платформах оркестровки контейнерів, таких як Kubernetes.

Клієнт MinIO надає альтернативу стандартним командам UNIX (наприклад, ls, cat, cp, mirror, diff тощо) і додає підтримку хмарних служб зберігання, сумісних з Amazon S3. Він працює на платформах Linux, Mac і Windows.

MinIO Client SDK надає API для доступу до будь-якого сервера зберігання об’єктів, сумісного з Amazon S3. Мовні прив’язки доступні для Go, Java, Python, JavaScript, Haskell і мов, розміщених поверх .NET Framework.

1.7 Безпекові протоколи TSL

TLS (англ. Transport Layer Security — захист на транспортному рівні), як і його попередник SSL — криптографічний протокол, що надає можливості безпечної передачі даних в інтернеті для навігації, отримання пошти, спілкування, обміну файлами, тощо. Використовує асиметричне шифрування і сертифікати X.509.

TLS надає можливості автентифікації і безпечної передачі даних через інтернет з використанням криптографічних засобів. Часто відбувається лише автентифікація сервера, а клієнт залишається неавтентифікованим. Для взаємної автентифікації кожна з сторін мусить підтримувати інфраструктуру відкритих ключів (PKI, англ. public key infrastructure), яка дозволяє захистити клієнт-серверні додатки від перехоплення, редагування повідомлень або ж створення підроблених.

TSL включає три основні фази:

- Діалог між сторонами, метою якого є вибір алгоритму шифрування.
- Обмін ключами на основі криптосистем з відкритим ключем або ж автентифікація на основі сертифікатів.
- Передача даних, що шифруються за допомогою симетричних алгоритмів шифрування.

Можуть бути доступні такі алгоритми:

- Для обміну ключами і перевірки їх справжності використовують комбінації алгоритмів: RSA (асиметричний шифр), Diffie-Hellman (безпечний обмін ключами), DSA (алгоритм цифрового підпису) і алгоритми технології Fortezza.
- Для симетричного шифрування: RC2, RC4, IDEA, DES, Triple DES або AES;
- Для хеш-функцій: MD5 або SHA.

Алгоритми можуть доповнюватися в залежності від версії протоколу.

Було доведено, що сучасні атаки на RC4 дозволяють зламати його протягом декількох днів або навіть годин. Тому в лютому 2015 року Internet Engineering Task Force (IETF) запропонувала в документі RFC 7465 припинити застосування RC4 в протоколі та реалізаціях TLS.

В серпні 2016 року в оновленні KB3151631 компанія Microsoft заявила, що припиняє використання RC4 в інтернет-браузерах починаючи з Internet Explorer 11 та Microsoft Edge.

РОЗДІЛ 2. НАЛАШТУВАННЯ ІНФРАСТРУКТУРИ ХМАРНОГО СЕРЕДОВИЩА

2.1 Встановлення операційної системи

Для подальшої роботи буде використана операційна система Ubuntu Server 20.04. Ubuntu Server — це варіант стандартного Ubuntu, призначений для мереж і служб. Він так само здатний запускати простий файловий сервер, як і працювати в хмарі з 50 000 вузлів.

На відміну від встановлення Ubuntu Desktop, Ubuntu Server не включає графічну програму встановлення. Замість цього він використовує процес на основі текстового меню.

Перед початком встановлення потрібно врахувати наступне:

- В наявності принаймні 2 ГБ вільного місця для зберігання.
- Мати доступ до DVD-диска або USB-накопичувача, що містить версію Ubuntu Server, яку потрібно встановити.
- При інстяляції Ubuntu Server на сховище з даними, які треба зберегти, спочатку зробити резервну копію.

Щоб почати процес інсталяції, потрібно виконати наступне:

1. Встановити диск з Ubuntu у DVD-дисковод (або вставити USB накопичувач чи іншу медіа).
2. Перезапустити комп’ютер.

Більшість комп’ютерів автоматично завантажуються з USB або DVD, хоча в деяких випадках це вимкнено, щоб скоротити час завантаження. Якщо ви не бачите повідомлення про завантаження та екран «Ласкаво просимо», який має з’явитися після нього, вам потрібно буде налаштувати комп’ютер на завантаження з інсталяційного носія.

На екрані має з’явитися повідомлення, коли комп’ютер почне повідомляти вам, яку клавішу натиснути для налаштувань або меню

завантаження. Залежно від виробника це може бути Escape, F2, F10 або F12. Просто перезавантажте комп’ютер і утримуйте цю клавішу, доки не з’явиться меню завантаження, а потім виберіть диск з інсталяційним носієм Ubuntu.

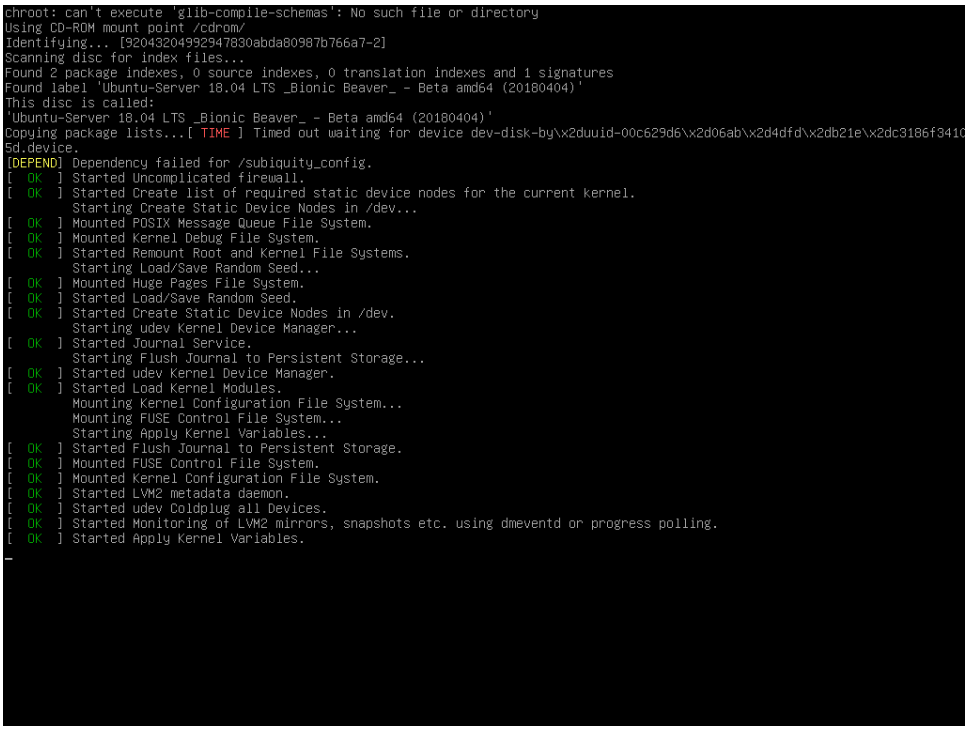


Рис. 2.1 Завантаження інтерфейсу інсталяції

З початку, користувачеві запропонують обрати мову операційної системи. За відсутності української мови, буде обрано англійську:

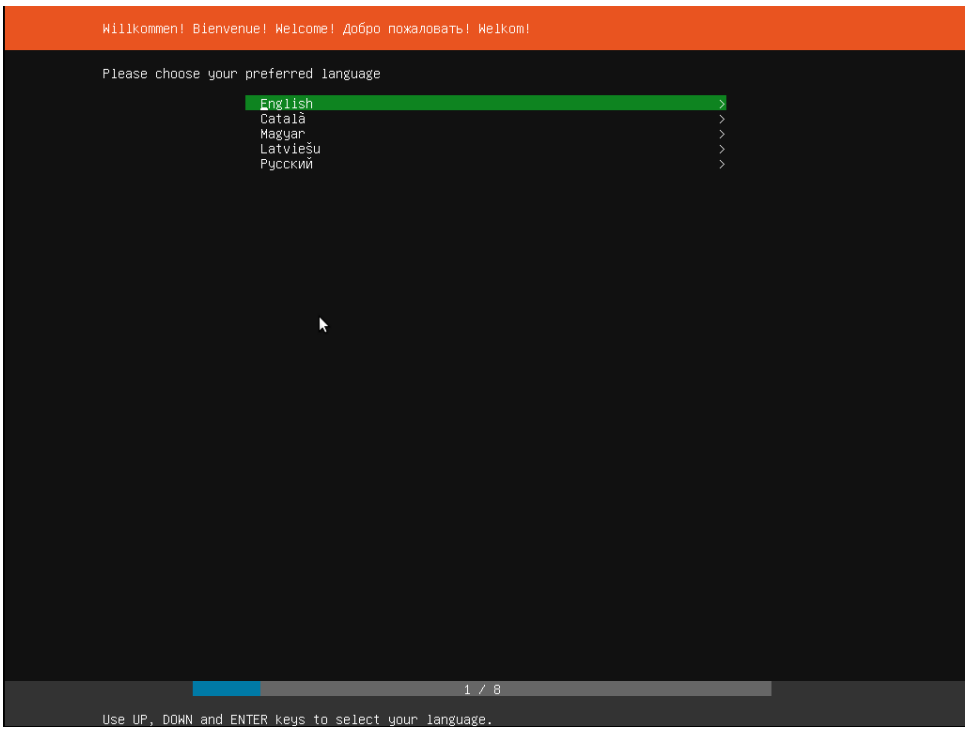


Рис. 2.2 Вибір мови операційної системи

Перш ніж вам потрібно буде щось ввести, інсталятор відобразить меню з проханням вибрати розкладку клавіатури та, якщо можливо, варіант.

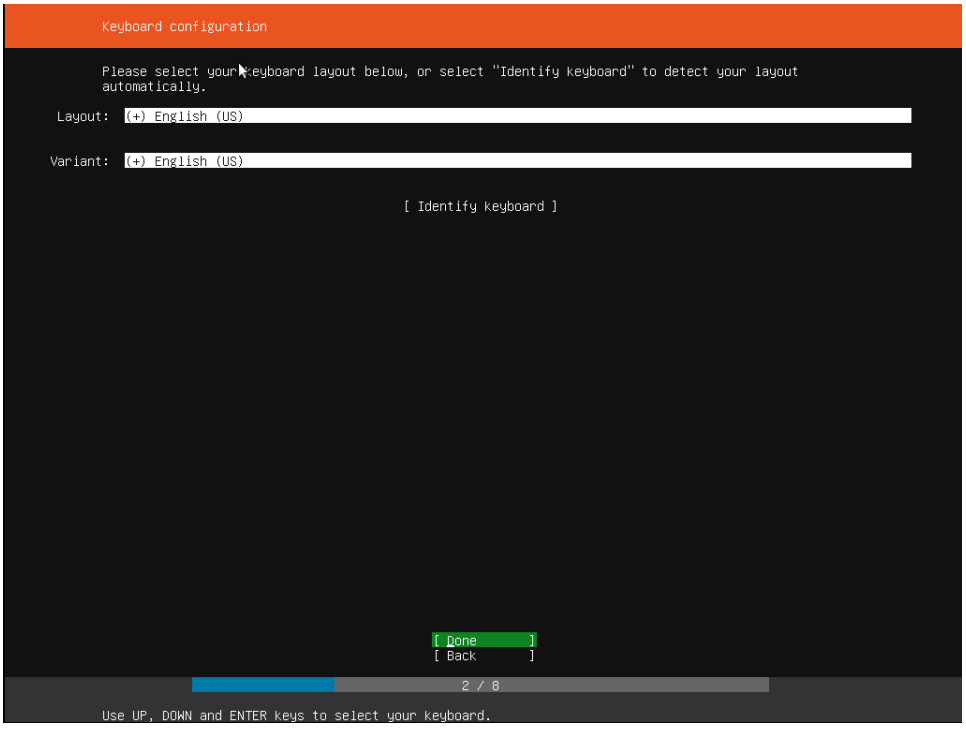


Рис. 2.3. Вибір розкладки та варіанта клавіатури

Тепер ми готові вибрати те, що ви хочете встановити. У меню є три варіанти, але нам потрібен Install Ubuntu.

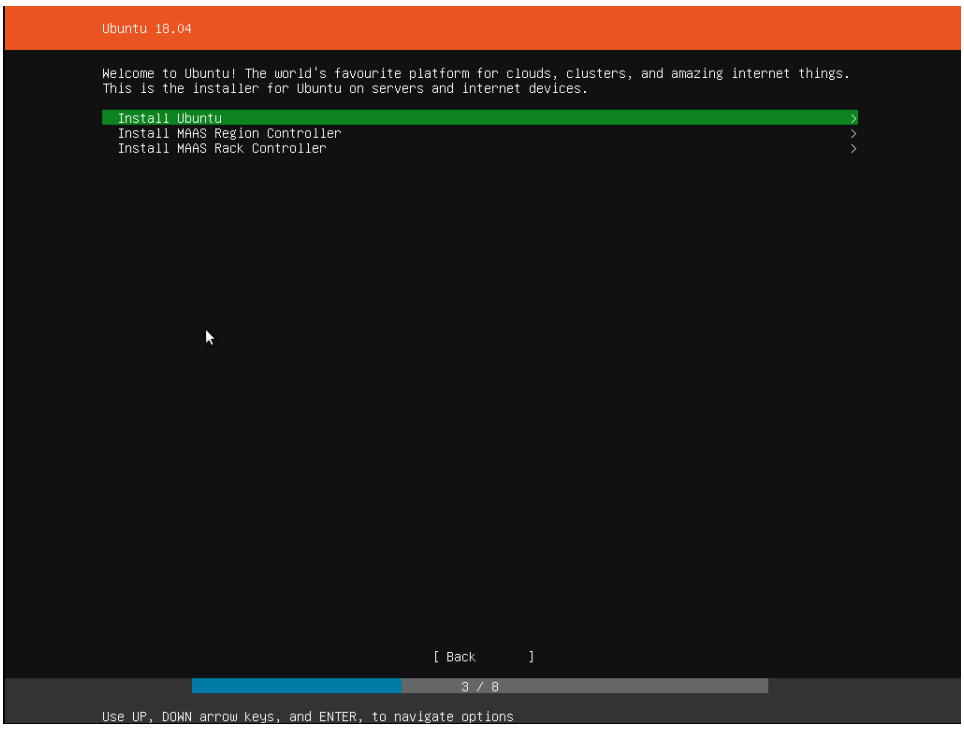


Рис. 2.4. Вибір компоненту для інсталяції

Інсталятор автоматично виявить і спробує налаштувати будь-які мережеві підключення через DHCP.

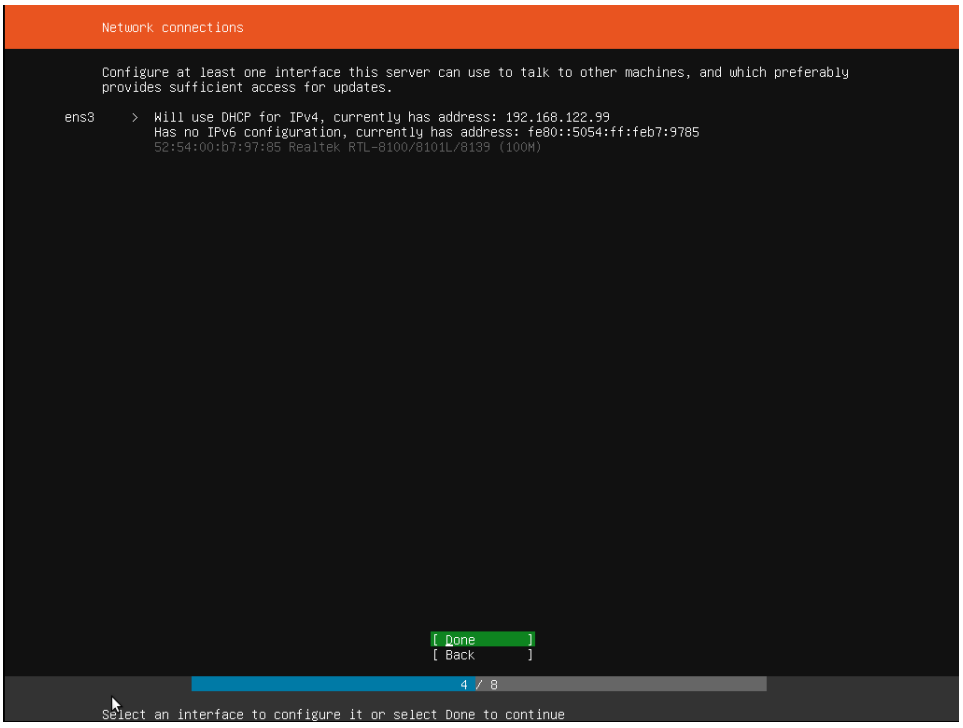


Рис. 2.5. Налаштування мережевого з’єднання

Наступним кроком буде налаштування сховища. Рекомендовано встановити весь диск або розділ для запуску Ubuntu.

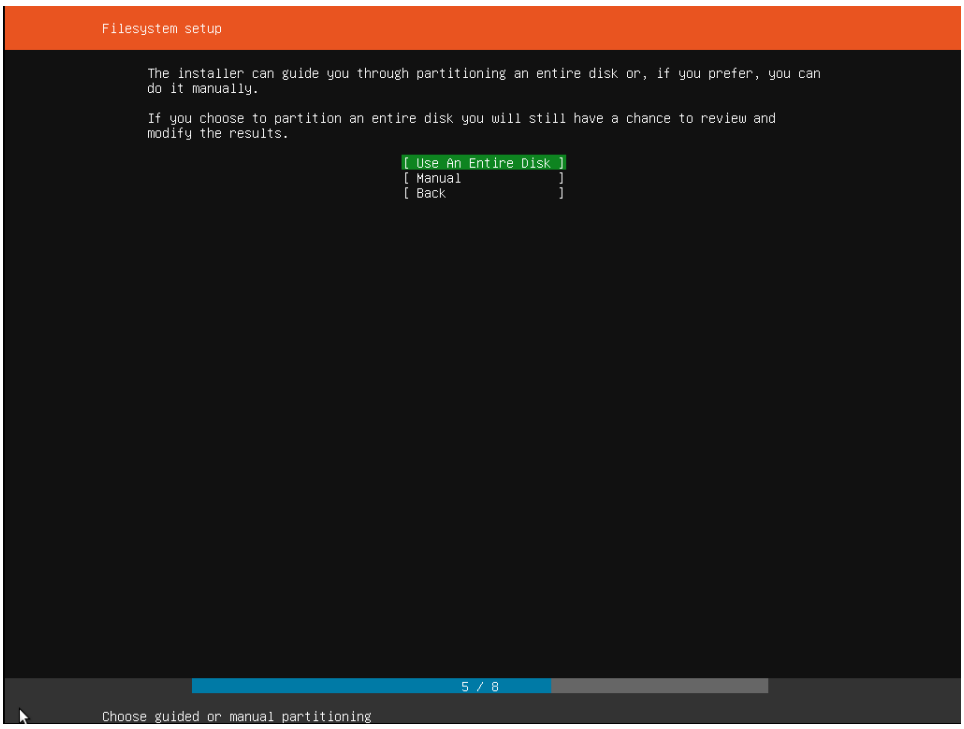


Рис. 2.6. Конфігурація сховища операційної системи

Вибравши цільовий диск, інсталятор обчислить, які розділи потрібно створити, і представить цю інформацію.

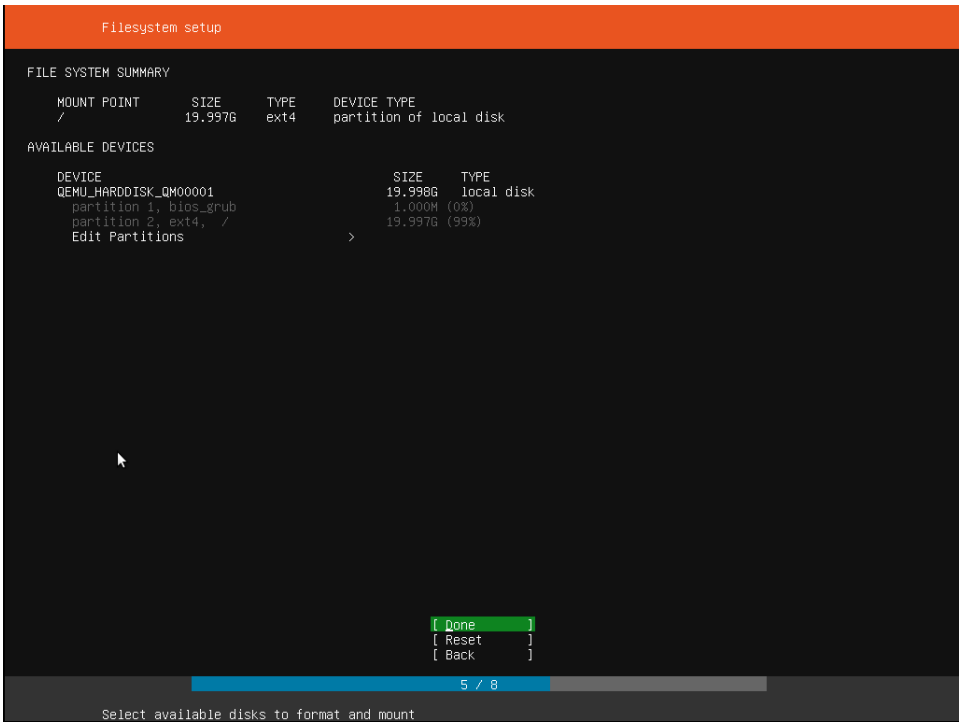
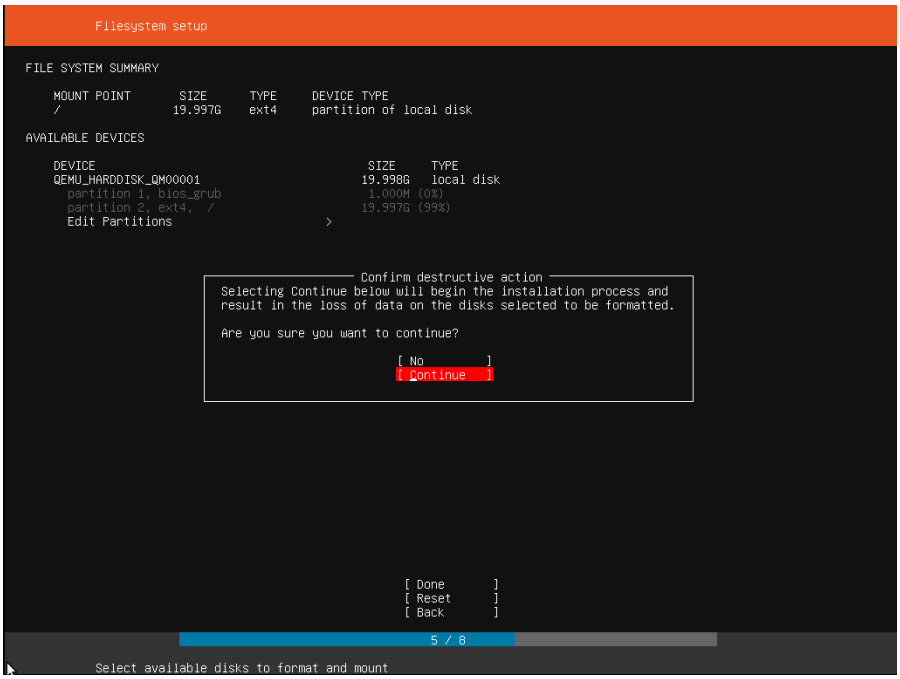


Рис. 2.7. Інформація про файлову систему

Перш ніж інсталятор внесе будь-які деструктивні зміни, він покаже цей останній крок підтвердження. Ще раз перевірте, чи все тут добре, і ви не



збираєтеся переформатувати неправильний пристрій!

Рис. 2.8. Попередження про форматування дискового носія

Програмне забезпечення зараз інсталюється на диск, але програмі встановлення потрібна додаткова інформація. Сервер Ubuntu повинен мати принаймні одного відомого користувача системи та ім’я хоста. Користувачеві також потрібен пароль.

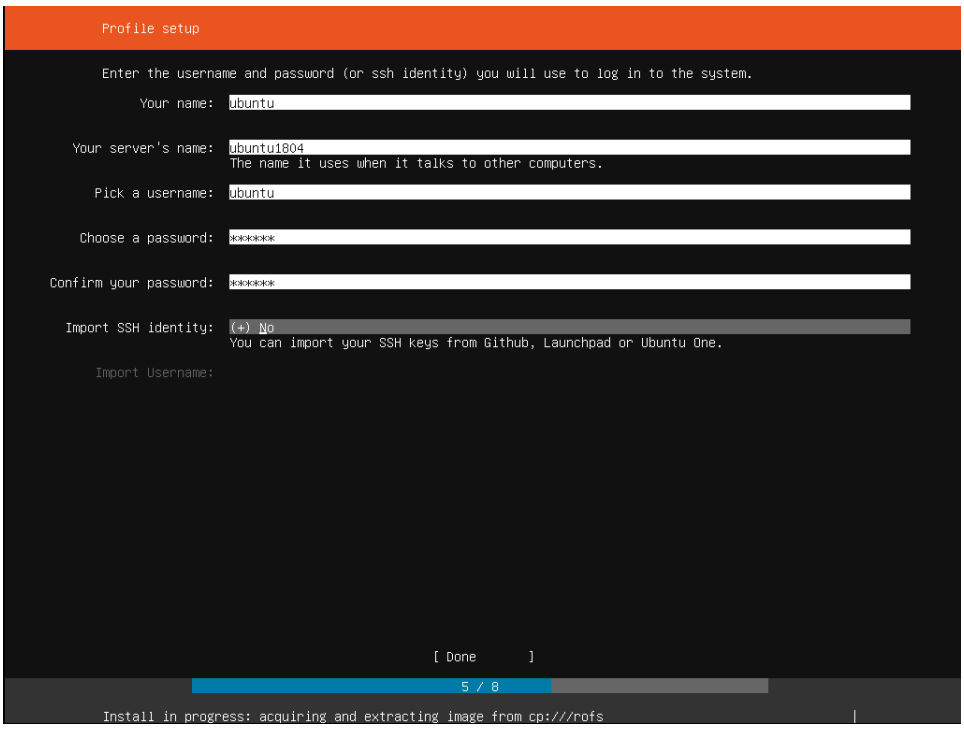


Рис. 2.9. Налаштування профілю користувача

Після того, як ви закінчите вводити необхідну інформацію, на екрані відображатиметься хід інсталятора. Ubuntu Server тепер інсталює короткий набір корисного програмного забезпечення, необхідного для серверів. Це значно скорочує час встановлення та налаштування. Звичайно, після завершення встановлення ви можете встановити будь-яке додаткове програмне забезпечення, яке може знадобитися.

Після завершення встановлення ви побачите на екрані таке повідомлення.

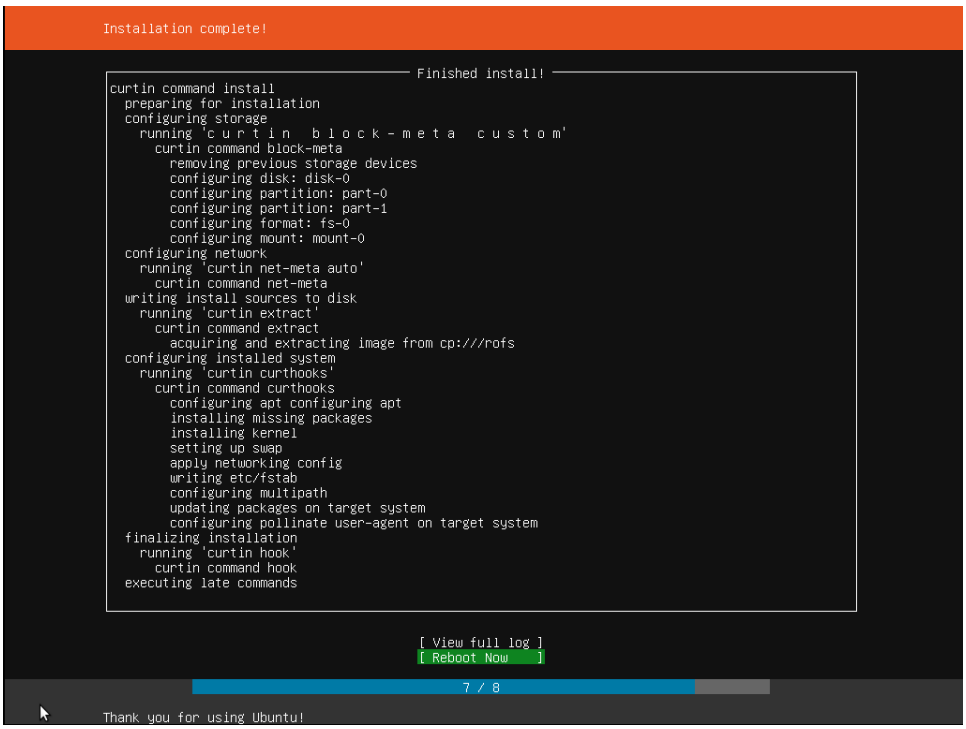


Рис. 2.10. Завершення інсталяції

Не забудьте видалити інсталяційний носій, а потім натисніть Enter, щоб перезавантажити та запустити сервер.

2.2 Встановлення веб-сервера

Веб-сервером для хмарного сховища буде виступати Nginx Web Server. Методом інсталяції буде поетапне введення команд у середі Ubuntu за допомогою репозиторію пакетів APT.

Спочатку, встановлюються пререквізити:

```
sudo apt install curl gnupg2 ca-certificates lsb-release ubuntu-keyring
```

Імпортуйте офіційний ключ підпису nginx, щоб apt міг перевірити автентичність пакетів. Отримати ключ:

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
    | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

Переконайтеся, що завантажений файл містить відповідний ключ:

```
gpg --dry-run --quiet --no-keyring --import --import-options import-show
/usr/share/keyrings/nginx-archive-keyring.gpg
```

Вихідні дані мають містити повний відбиток 573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62 таким чином:

```
pub   rsa2048 2011-08-19 [SC] [expires: 2024-06-14]
      573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62
uid                 nginx signing key signing-key@nginx.com
```

Якщо відбиток інший, видаліть файл.

Щоб налаштувати репозиторій apt для стабільних пакетів nginx, виконайте таку команду:

```
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/ubuntu `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

Налаштуйте закріплення репозиторію, щоб надавати перевагу нашим пакетам над пакетами, що надаються дистрибутивом:

```
echo -e "Package: *\nPin: origin nginx.org\nPin: release o=nginx\nPin-Priority:
900\n" \
    | sudo tee /etc/apt/preferences.d/99nginx
```

Щоб встановити Nginx, виконайте такі команди:

```
sudo apt update
sudo apt install nginx
```

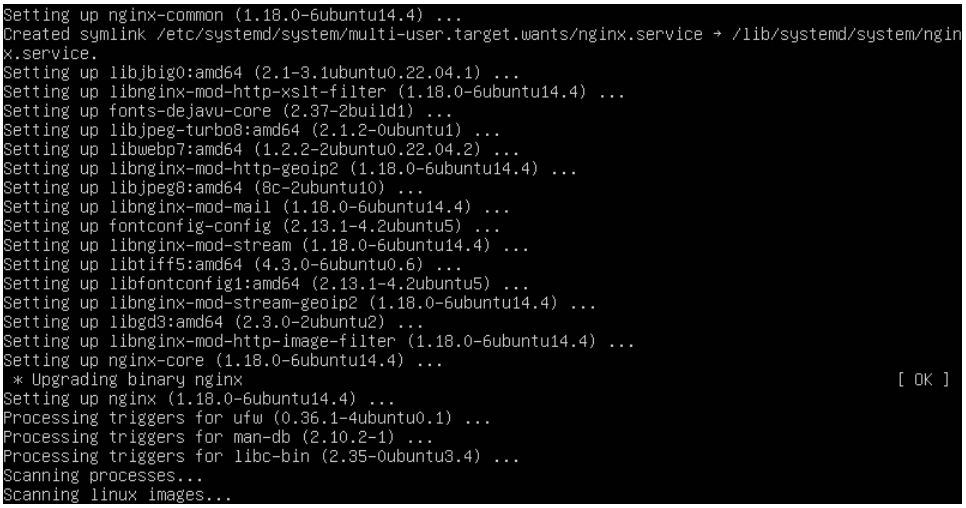


Рис. 2.11. Завершення інсталяції Nginx

2.3 Встановлення сервера баз даних MySQL

На Ubuntu 20.04 ви можете встановити MySQL за допомогою репозиторію пакетів APT. На момент написання цієї статті версією MySQL, доступною в репозиторії Ubuntu за замовчуванням, є версія 8.0.27.

Щоб установити його, оновіть індекс пакетів на своєму сервері, якщо ви не робили цього нещодавно:

```
sudo apt update
```

Потім встановіть пакет mysql-server:

```
sudo apt install mysql-server
```

Переконайтеся, що сервер працює за допомогою команди systemctl start:

```
sudo systemctl start mysql.service
```

Ці команди встановлять і запустять MySQL, але не запропонують вам встановити пароль або внести будь-які інші зміни конфігурації. Оскільки це робить вашу установку MySQL небезпечною, ми розглянемо це далі.

Запустіть сценарій безпеки за допомогою sudo:

```
sudo mysql_secure_installation
```

Це проведе вас через серію підказок, де ви можете внести деякі зміни в параметри безпеки вашої установки MySQL. Перше підказка запитає, чи хочете ви налаштувати плагін перевірки пароля, який можна використовувати для перевірки надійності паролів нових користувачів MySQL, перш ніж вважати їх дійсними.

Якщо ви вирішите налаштувати модуль Validate Password Plugin, будь-який створений вами користувач MySQL, який автентифікується за допомогою пароля, повинен буде мати пароль, який відповідає вибраній вами політиці. Найсильніший рівень політики, який можна вибрати, ввівши 2, вимагатиме, щоб паролі мали довжину принаймні вісім символів і містили комбінацію великих і малих літер, цифр і спеціальних символів:

```
Output
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?
```

Press y|Y for Yes, any other key for No: Y

There are three levels of password validation policy:

LOW Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG:
2

Незалежно від того, чи ви вирішите налаштувати плагін перевірки пароля, наступним запитом буде встановлення пароля для користувача root MySQL. Введіть і підтвердіть безпечний пароль на свій вибір:

Output
Please set the password for root here.

New password:

Re-enter new password:

Зауважте, що навіть якщо ви встановили пароль для користувача root MySQL, цей користувач наразі не налаштований на автентифікацію за допомогою пароля під час підключення до оболонки MySQL.

Якщо ви використовували плагін перевірки пароля, ви отримаєте відгук про надійність нового пароля. Тоді сценарій запитає, чи бажаєте ви продовжити з паролем, який ви щойно ввели, чи бажаєте ввести новий. Якщо

ви задоволені надійністю щойно введеного пароля, введіть Y, щоб продовжити сценарій:

Output

Estimated strength of the password: 100

Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : Y

Звідти ви можете натиснути Y, а потім ENTER, щоб прийняти значення за замовчуванням для всіх наступних запитань. Це видалить деяких анонімних користувачів і тестову базу даних, вимкне віддалені кореневі входи та завантажить ці нові правила, щоб MySQL негайно поважав внесені вами зміни.

Після завершення сценарію ваша інсталяція MySQL буде захищена. Тепер ви можете перейти до створення спеціального користувача бази даних за допомогою клієнта MySQL.

Після встановлення MySQL створює обліковий запис користувача root, який можна використовувати для керування базою даних. Цей користувач має повні привілеї щодо сервера MySQL, тобто він має повний контроль над кожною базою даних, таблицею, користувачем тощо. Через це краще уникати використання цього облікового запису поза адміністративними функціями. Цей крок описує, як використовувати користувача root MySQL для створення нового облікового запису користувача та надання йому привілеїв.

У системах Ubuntu, на яких працює MySQL 5.7 (і пізніші версії), користувач root MySQL налаштований на автентифікацію за допомогою плагіна auth_socket за замовчуванням, а не за допомогою пароля. Цей плагін вимагає, щоб ім’я користувача операційної системи, який викликає клієнт MySQL, відповідало імені користувача MySQL, указаному в команді, тому ви повинні викликати mysql з привілеями sudo, щоб отримати доступ до кореневого користувача MySQL:

sudo mysql

Отримавши доступ до підказки MySQL, ви можете створити нового користувача за допомогою оператора CREATE USER. Вони дотримуються такого загального синтаксису:

mysql> CREATE USER 'username'@'host' IDENTIFIED WITH authentication_plugin BY 'password';

Після CREATE USER ви вказуєте ім’я користувача. Одразу слідує знак @, а потім ім’я хоста, з якого підключатиметься цей користувач. Якщо ви плануєте отримати доступ до цього користувача лише локально з вашого сервера Ubuntu, ви можете вказати localhost. Брати ім’я користувача та хост в одинарні лапки не завжди потрібно, але це може допомогти запобігти помилкам.

У вас є кілька варіантів, коли справа доходить до вибору плагіна автентифікації вашого користувача. Згаданий вище плагін auth_socket може бути зручним, оскільки він забезпечує надійний захист, не вимагаючи від дійсних користувачів вводити пароль для доступу до бази даних. Але це також запобігає віддаленим підключенням, що може ускладнити ситуацію, коли зовнішні програми повинні взаємодіяти з MySQL.

Виконайте таку команду, щоб створити користувача, який автентифікується за допомогою caching_sha2_password:

mysql> CREATE USER 'delight'@'localhost' IDENTIFIED BY 'neptune@4221';

Після створення нового користувача ви можете надати йому відповідні привілеї. Загальний синтаксис для надання привілеїв користувача такий:

```
mysql>  GRANT PRIVILEGE ON database.table TO 'username'@'host';
```

Значення PRIVILEGE у цьому прикладі синтаксису визначає дії, які користувач може виконувати з указаною базою даних і таблицею. Ви можете надати кілька привілеїв одному користувачеві в одній команді, розділяючи їх комами. Ви також можете надати привілеї користувача глобально, ввівши зірочки (*) замість імен бази даних і таблиць. У SQL зірочки — це спеціальні символи, які використовуються для представлення «всіх» баз даних або таблиць.

Для прикладу: наступна команда надає користувачеві глобальні права СТВОРЮВАТИ, ЗМІНЮВАТИ та ВИДАЛЯТИ бази даних, таблиць і користувачів, а також право ВСТАВЛЯТИ, ОНОВЛЯТИ та ВИДАЛЯТИ дані з будь-якої таблиці на сервері. Він також надає користувачеві можливість запитувати дані за допомогою SELECT, створювати зовнішні ключі за допомогою ключового слова REFERENCES і виконувати операції FLUSH з привілеєм RELOAD. Однак ви повинні надавати користувачам лише ті дозволи, які їм потрібні, тому не соромтеся змінювати привілеї своїх власних користувачів за потреби.

```
mysql>  GRANT CREATE, ALTER, DROP, INSERT, UPDATE, INDEX, DELETE, SELECT, REFERENCES, RELOAD on *.* TO 'delight'@'localhost' WITH GRANT OPTION;
```

Після цього доцільно виконати команду FLUSH PRIVILEGES. Це звільнить будь-яку пам’ять, яку сервер кешував у результаті попередніх операторів CREATE USER і GRANT:

mysql> FLUSH PRIVILEGES;

Потім ви можете вийти з клієнта MySQL:

mysql> exit

У майбутньому, щоб увійти як ваш новий користувач MySQL, ви будете використовувати таку команду:

mysql -u delight -p

Прапор -p призведе до того, що клієнт MySQL запропонує вам ввести пароль користувача MySQL для автентифікації.

Незалежно від того, як ви його встановили, MySQL мав почати працювати автоматично. Щоб перевірити це, перевірте його статус.

systemctl status mysql.service

Ви побачите результат, схожий на такий:

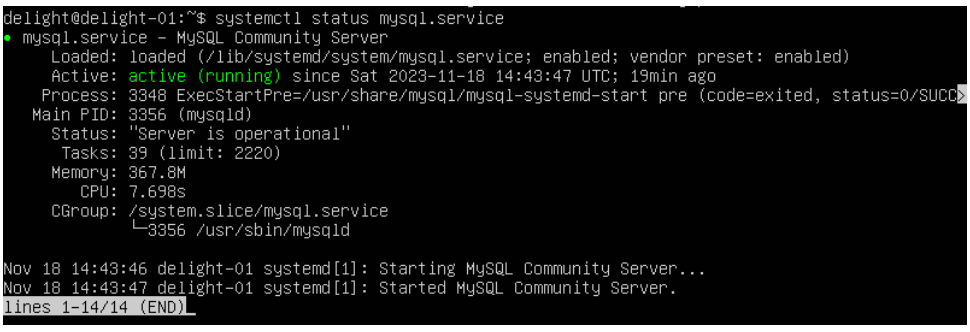


Рис. 2.12. Звіт служби MySQL

Якщо MySQL не запущено, ви можете запустити його за допомогою sudo systemctl start mysql.

2.4 Встановлення сховища об’єктів MinIO

Наступні команди забезпечують інсталяцію MinIO у 64-розрядних операційних системах Linux за допомогою RPM, DEB або двійкового коду. Пакети RPM і DEB автоматично встановлюють MinIO до необхідних системних шляхів і створюють службу minio для systemctl.

Використовуйте такі команди, щоб завантажити останню стабільну версію MinIO DEB і встановити її:

```
wget https://dl.min.io/server/minio/release/linux-amd64/archive/minio_20231120224007.0.0_amd64.deb -O minio.deb
sudo dpkg -i minio.deb
```

Виконайте наступну команду з системного терміналу або оболонки, щоб запустити локальний екземпляр MinIO за допомогою папки ~/minio. Ви можете замінити цей шлях на інший шлях до папки на локальній машині:

```
mkdir ~/minio
minio server ~/minio --console-address :9090
```

Команда mkdir явно створює папку за вказаним шляхом.

Команда minio server запускає сервер MinIO. Аргумент шляху ~/minio визначає папку, в якій працює сервер.

Серверний процес minio друкує вихідні дані на системну консоль, подібно до такого:

Рис. 2.13 Звіт служби MinIO

Відкрийте http://127.0.0.1:9000 у веб-браузері, щоб отримати доступ до консолі MinIO. Ви також можете ввести будь-яку з мережевих адрес, указаних у вихідних даних команди сервера. Наприклад, консоль: http://10.0.2.15:9090 http://127.0.0.1:9090 у вихідних даних прикладу вказує на дві можливі адреси для підключення до консолі.

Хоча порт 9000 використовується для підключення до API, MinIO автоматично перенаправляє доступ браузера до консолі MinIO.

Увійдіть до консолі за допомогою облікових даних користувача RootUser і RootPass, які відображаються у вихідних даних. За замовчуванням це minioadmin | minioadmin.

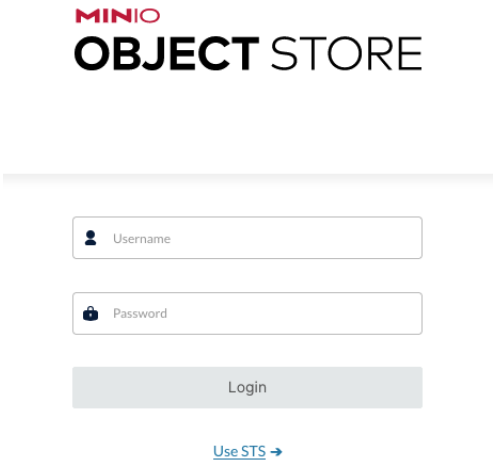


Рис. 2.14. Вхід до консолі MinIO

Ви можете використовувати консоль MinIO для загальних завдань адміністрування, таких як керування ідентифікацією та доступом, моніторинг показників і журналів або конфігурація сервера. Кожен сервер MinIO містить власну вбудовану консоль MinIO.

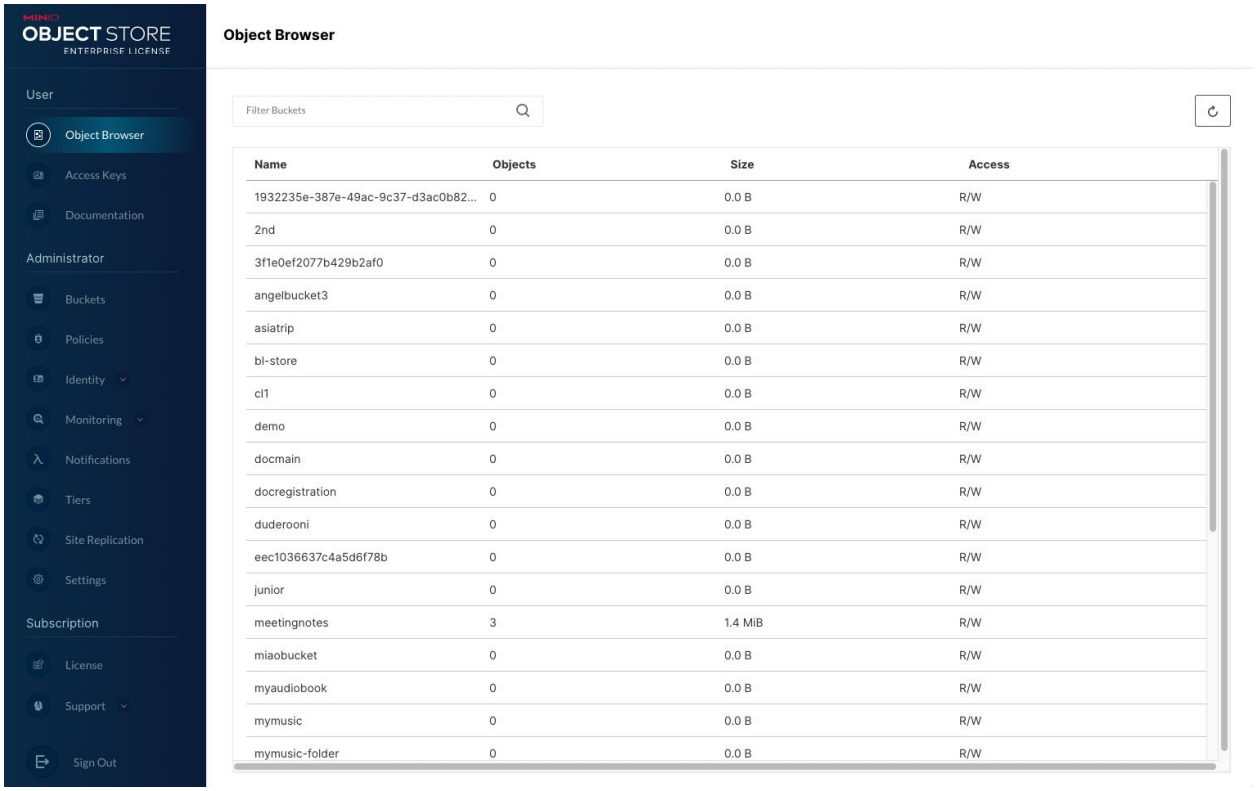


Рис. 2.15. Браузер об’єктів в консолі MinIO

2.5 Створення інтерфейсу менеджеру хмарного сховища

Розробка фронтенду для менеджменту файлів та папок у хмарному сховище включає створення інтерфейсу користувача, що дозволяє взаємодіяти з бекенд-службами – наприклад, сховищем об’єктів. Нижче буде представлений базовий веб-фронтенд з використанням HTML, CSS та JavaScript.

Програмний код буде прикріплений у додатках А-В.

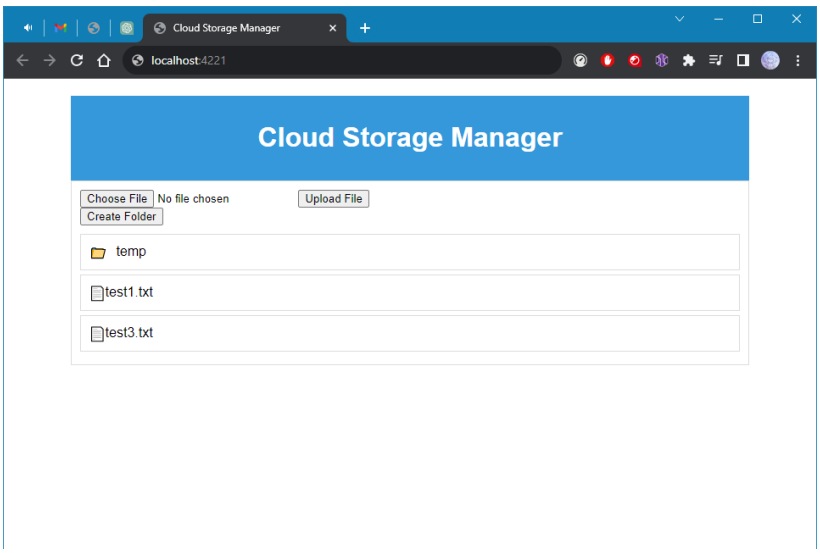


Рис. 2.16. Простий інтерфейс менеджера файлів

Завантаження файлів у JavaScript буде виконуватися за допомогою елементу HTML `<input type=”file”>` разом із JavaScript, щоб виконувати процеси обрання файлів та їх завантаження. Користувач може обрати файл використовуючи елемент вводу файлів, а функція `uploadFile` буде виконуватися коли буде натиснута кнопка “Upload File”. Обраний файл буде додано до об’єкту `FormData`, симулюючи відправку форми.

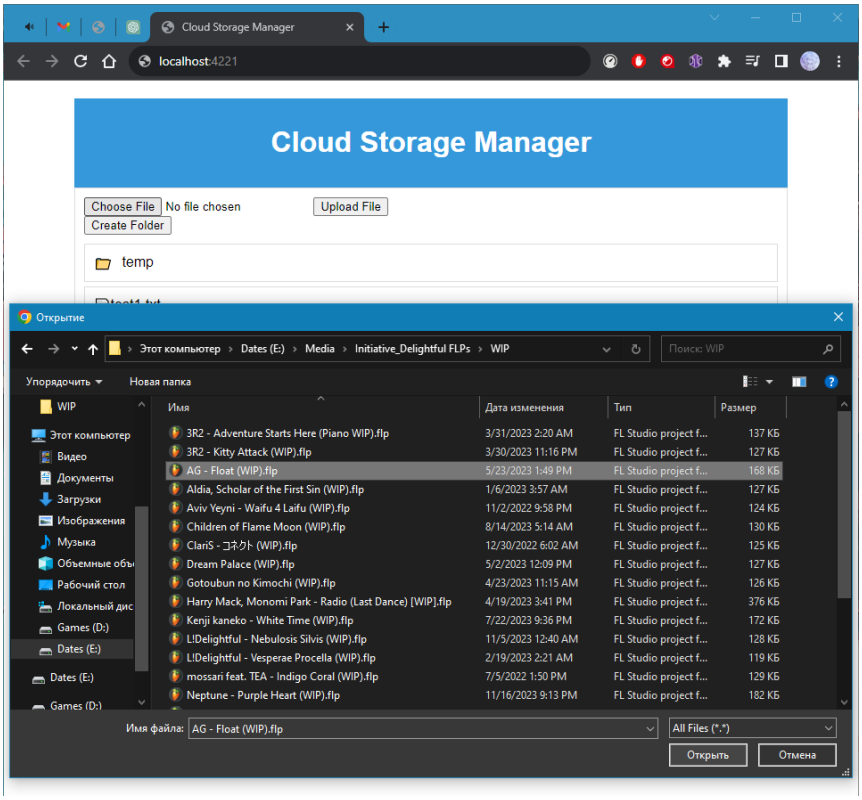


Рис. 2.17. Обрання бажаного файлу для завантаження

Механізм завантаження файлів було реалізовано за допомогою Javascript об’єкту `XMLHttpRequest`. Він дозволяє відправляти запроси HTTP або HTTPS на веб-сервер та отримувати дані з серверу без необхідності повністю перезавантажувати веб-сторінку. `XMLHttpRequest` є головною частиною AJAX (Asynchronous JavaScript and XML) та зазвичай використовується для асинхронної передачі даних.

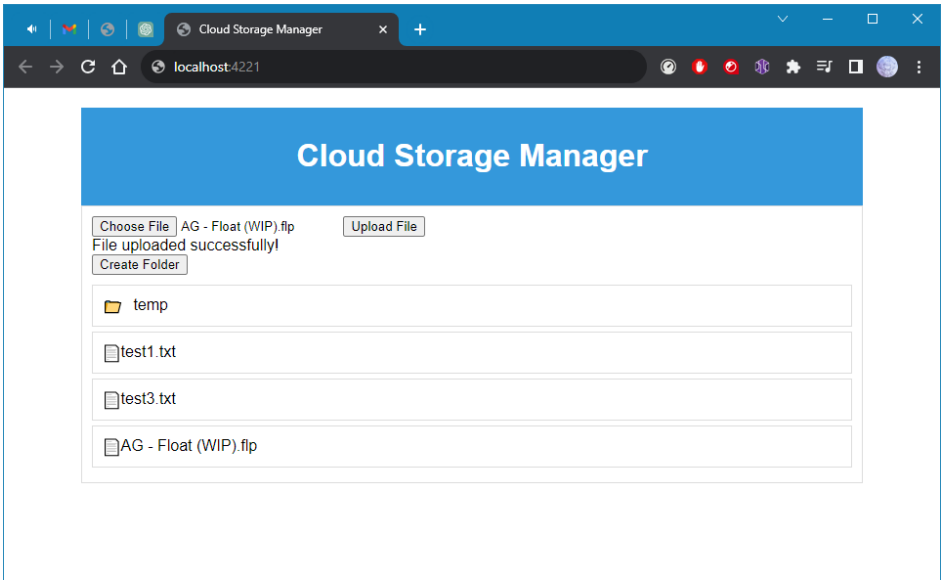


Рис. 2.18. Успішно завантажений файл

ВИСНОВКИ

1. У ході виконання дипломної роботи проведено аналітичний догляд і розглянуті різні методи створення хмарного сховища.
2. В середовищі операційної системи Ubuntu Server 20.04 було створено інфраструктуру хмарного сховища файлів. До неї входили: веб-сервер Nginx, сервер баз даних MySQL та сховище об'єктів MinIO. Для подальшої роботи з сервісом, потребується цілодобове обслуговування сервера та виконання усіх безпекових вимог для захисту інформації.
3. Сховищем файлів виступав бакет у середі сховища MinIO, що дозволило використовувати його у механізмі завантаження файлів. Під час роботи було використано локальну адресу сервера, що полегшило тестування програмного забезпечення та прискорило його розробку. Для подальшого використання користувачем рекомендується використовувати хостинг для доступу до сервісів.
4. Отримані результати роботи й розроблене програмне забезпечення здатне зберігати інформацію, наприклад фото, відео, документи та ін. у вигляді даних у віддаленому хмарному сховищі. Масштабність системи залежить від технічних обмежень. Для тестування було використано бакет розміром у 2 ГБ.

Подібний комплекс може використовуватися у якості бюджетної версії хмарного сховища.

СПИСОК ЛІТЕРАТУРИ

1. "The Big Switch: Rewiring the World, from Edison to Google" by Nicholas Carr.
2. "Cloud Computing: Concepts, Technology & Architecture" by Thomas Erl, Zaigham Mahmood, and Ricardo Puttini.
3. "Architecting the Cloud: Design Decisions for Cloud Computing Service Models" by Michael J. Kavis.
4. "Cloud Native Patterns: Designing Change-tolerant Software" by Cornelia Davis.
5. "The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win" by Gene Kim, Kevin Behr, and George Spafford.
6. "Cloud Computing: A Hands-On Approach" by Arshdeep Bahga and Vijay Madisetti.
7. "Google Cloud Platform in Action" by J.D. Velásquez and M. Tim Jones.

ДОДАТОК А

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cloud Storage Manager</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="app">
        <div id="header">
            <h1>Cloud Storage Manager</h1>
        </div>
        <div id="file-manager">
            <div id="toolbar">
                                    <input type="file" id="fileInput" />
                <button onclick="uploadFile()">Upload File</button>
                                    <div id="uploadStatus"></div>

                <button onclick="createFolder()">Create Folder</button>
            </div>
            <div id="file-list">
                <!-- File and folder list will be displayed here -->
            </div>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

ДОДАТОК Б

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

#app {
    max-width: 800px;
    margin: 20px auto;
}

#header {
    background-color: #3498db;
    color: #fff;
    padding: 10px;
    text-align: center;
}

#file-manager {
    border: 1px solid #ddd;
    padding: 10px;
}

#toolbar {
    margin-bottom: 10px;
}

#file-list {
    list-style-type: none;
    padding: 0;
}

.file-item {
    border: 1px solid #ddd;
    margin-bottom: 5px;
    padding: 10px;
    display: flex;
    align-items: center;
}
```

```
.folder-icon {
    margin-right: 10px;
    color: #3498db;
}
```

ДОДАТОК В

```
// Function to render file and folder list
function renderFileList() {
    const fileListElement = document.getElementById('file-list');
    fileListElement.innerHTML = '';

    filesAndFolders.forEach(item => {
        const listItem = document.createElement('li');
        listItem.className = 'file-item';

        const icon = document.createElement('span');
        icon.className = item.type === 'file' ? 'file-icon' : 'folder-icon';
        icon.innerHTML = item.type === 'file' ? '' : '';

        const name = document.createElement('span');
        name.textContent = item.name;

        listItem.appendChild(icon);
        listItem.appendChild(name);
        fileListElement.appendChild(listItem);
    });
}

// Function to handle file upload
function uploadFile() {
    const fileInput = document.getElementById('fileInput');
    const uploadStatus = document.getElementById('uploadStatus');

    // Check if a file is selected
    if (fileInput.files.length === 0) {
        alert('Please select a file.');
        return;
    }

    // Get the selected file
    const file = fileInput.files[0];

    // Create a FormData object to send the file
    const formData = new FormData();
    formData.append('file', file);
```

```
    var xhr = new XMLHttpRequest();
        xhr.open('POST', 'https://localhost:9090/buckets/thesisTest1', true);
        xhr.onreadystatechange = function() {
                if (xhr.readyState === XMLHttpRequest.DONE) {
                        if (xhr.status === 200) {
                                console.log(xhr.responseText);
                        } else {
                                console.error('Request failed with status:', xhr.status);
                        }
                }
        };
        xhr.send();


    setTimeout(() => {
       uploadStatus.textContent = 'File uploaded successfully!';
    }, 1000);
}

// Function to handle folder creation (replace this with actual folder creation logic)
function createFolder() {
    alert('Implement folder creation functionality here.');
}

// Initial render
renderFileList();
```