МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

GRID-системи та хмарні технології

Методичні вказівки
до виконання лабораторних робіт
для здобувачів другого (магістерського) рівня вищої освіти
спеціальностей F2 (121) «Інженерія програмного забезпечення»,
F6 (126) «Інформаційні системи та технології»

Київ 2025

УДК 004.042
С60

Укладач О. Л. Соловей, канд. техн. наук

Рецензент І. А. Ачкасов, д-р техн. наук, професов

Відповідальна за випуск Т. А. Гончаренко, д-р техн. наук, професор

Затверджено на засіданні кафедри інформаційних технологій, протокол № 5 від 10 грудня 2025 року.

В авторській редакції.

GRID-системи та хмарні технології [електронний ресурс] :
С60 методичні вказівки до виконання практичних та лабораторних робіт / уклад. О. Л. Соловей. – Київ : КНУБА, 2025. – 36 с.

Містять теоретичні відомості і рекомендації щодо виконання лабораторних робіт з дисципліни та вимоги до оформлення звіту. Спрямовані на організацію самостійної роботи студентів.

Призначені для здобувачів другого (магістерського) рівня вищої освіти спеціальностей F2 (121) «Інженерія програмного забезпечення», F6 (126) «Інформаційні системи та технології».

Зміст

Загальні положення

Лабораторні роботи є логічним продовженням лекційного курсу з дисципліни “GRID-системи та хмарні технології” і є перехідною ланкою від теоретичного курсу до набуття практичних навичок роботи з хмарними технологіями платформи Azure.

Кожна лабораторна робота містить такі види робіт:

- аналіз умови задачі.
- виконання задача в хмарному середовищі Azure.
- демонстрацію виконаного завдання.
- відповіді на контрольні запитання.
- складання і захист звіту.

Лабораторна робота №1

Робота з Azure Blob Storage

Мета роботи: Набути навичок роботи з Azure Blob Storage.

Завдання

Частина 1. Створити статичний вебсайт за допомогою сховища Blob Storage..

Частина 2. Навчитися створювати та управляти контейнерами й blob-файлами в Azure Blob Storage, використовуючи класи BlobServiceClient, ContainerClient, BlobClient.

Теоретичні відомості

Azure Blob Storage — це рішення для зберігання великих обсягів неструктурованих даних у хмарі Microsoft Azure.

Воно дозволяє зберігати дані у вигляді об’єктів (blobs), таких як текстові або двійкові файли. Це популярне рішення для створення резервних копій, архівів, зберігання мультимедійних даних, а також для обробки аналітичних big data-навантажень.

Основні компоненти Azure Blob Storage:

1. Storage Account (обліковий запис зберігання) - Це логічна одиниця, яка надає доступ до служби Azure Storage.

Обліковий запис є контейнером для різних типів сховищ, включно з Blob Storage. Він може використовуватися для створення та керування різними типами сховищ даних: Blob, Queue, File і Table Storage. Забезпечує контроль доступу, моніторинг і керування збереженими даними.

2. Containers (контейнери) - Контейнер є логічною структурою в Blob Storage, що містить групу blob-об’єктів. Кожен обліковий запис зберігання може містити кілька контейнерів, а кожен контейнер — практично необмежену кількість об’єктів.

3. Blobs (об’єкти) - Це безпосередньо дані, що зберігаються в контейнері. Існує три типи blob-об’єктів:

Block blobs — зберігають текстові або двійкові дані; ідеально підходять для великих файлів, як-от документи чи мультимедіа.

Append blobs — різновид block blobs, оптимізований для послідовного дописування (наприклад, лог-файли).

Page blobs — зберігають дані у вигляді сторінок; призначені для сценаріїв із випадковим доступом, зокрема для віртуальних жорстких дисків (VHD).

Хід виконання роботи

1. Увійдіть на портал Azure.

2. На панелі «Усі служби» знайдіть і виберіть «Служба сховища» (Storage Account) та натисніть «Створити».

3. Вкажіть основні відомості для створення нової служби сховища (назва облікового запису сховища має містити ваше прізвище).У розділі «Безпека» (Security) виберіть "Enable storage account key access". У розділі «Мережі» (Network connectivity) виберіть "Enable public access from all networks"

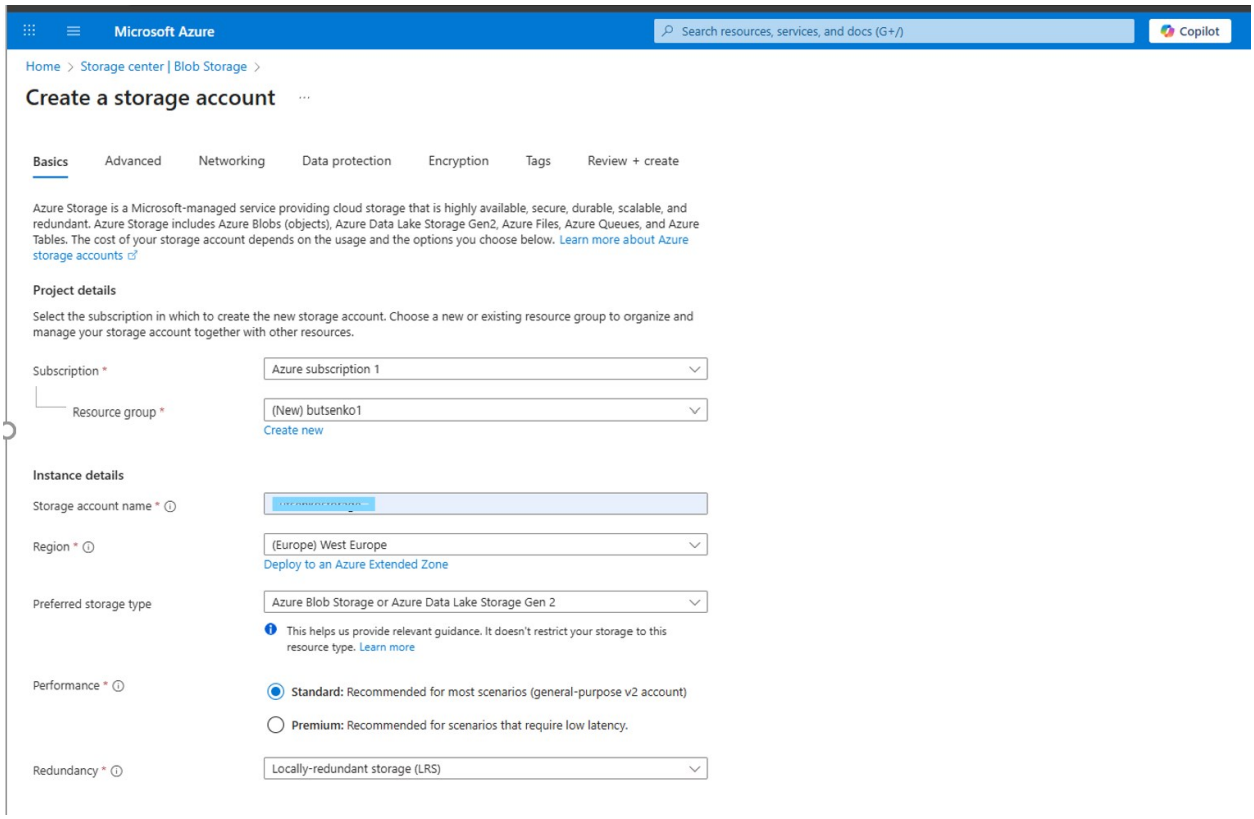


Рис. 1. Створення нової служби сховища

4. Натисніть «Переглянути та створити», щоб розпочати автоматичну перевірку. Після завершення розгортання натисніть «Перейти до ресурсу».

5. На локальному комп’ютері створіть домашню сторінку index.html для вашого статичного вебсайту. На сторінці має бути вказано ваше прізвище у нижній частині. Збережіть файл.

6. Перейдіть до розділу «Static website». Активуйте його. Введіть назву файлу index.html. Збережіть. У результаті буде згенеровано посилання на ваш вебсайт.

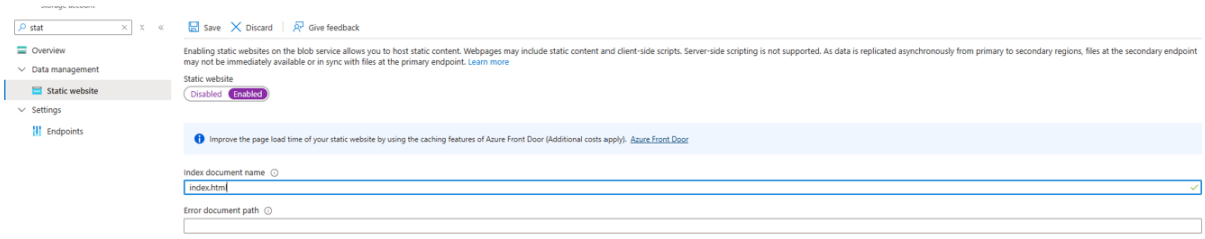


Рис. 2. Активація функції «Static website»

7. Перейдіть до меню «Storage Browser» → «Blob containers» та завантажте створений index.html у контейнер web.

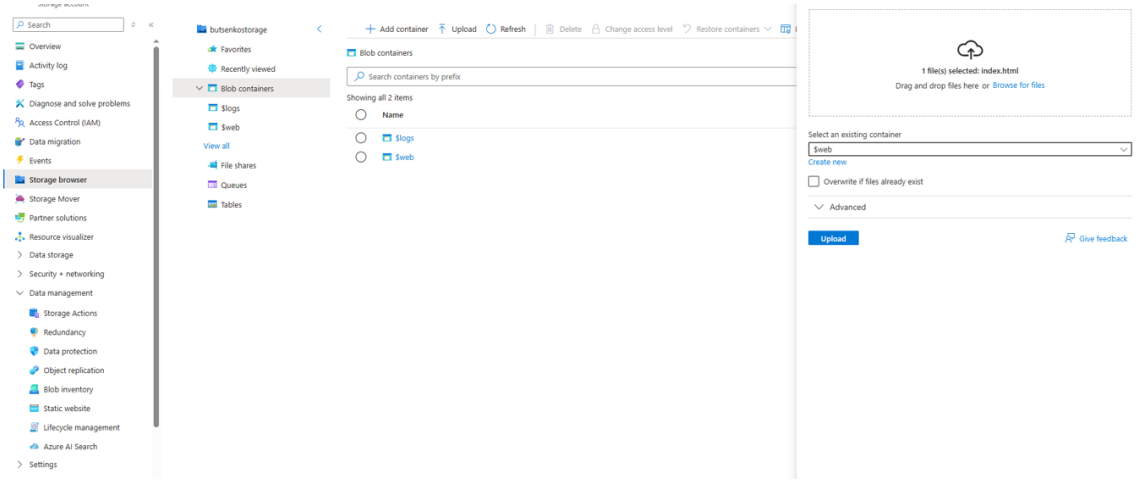


Рисунок 3. Завантаження index.html у контейнер web.

8. Скопіюйте посилання на вашу сторінку та відкрийте його в новому вікні браузера. У результаті ви маєте побачити завантажену домашню сторінку.

9. Мовою програмування Python підключіться до вашого облікового запису Azure Storage (ініціалізуйте об'єкт BlobServiceClient через connection string).

a. Виведіть список усіх контейнерів.
b. Створіть контейнер з унікальною назвою.
c. Перевірте, чи він з’явився у списку контейнерів.
d. Створіть локальний файл.
e. Завантажте його у створений контейнер за допомогою BlobClient.

f. Додайте до blob'a метадані.

10. Підготуйте звіт, який включає: знімки екрана з результатами виконання кроків 6–8; робочий код програми; відповіді на контрольні запитання.

Контрольні запитання

1. Для яких сценаріїв призначені Queue Storage та Table Storage?
2. Які типові бізнес-задачі можна вирішити за допомогою Block blobs, Append blobs та Page blobs?
3. У чому полягає різниця між локально надлишковим сховищем (LRS) та зонально надлишковим сховищем (ZRS)?
4. Які переваги надає географічно надлишкове сховище (GRS) порівняно з LRS та ZRS?
5. Чому варто враховувати затримки та вартість при виборі варіанту копіювання даних між регіонами?
6. У яких випадках GRS може бути критично важливим для бізнесу?
7. Чим відрізняються рівні доступу Hot, Cool і Archive за вартістю та продуктивністю?
8. Які критерії використовуються для вибору відповідного рівня доступу до даних?
9. Що відбувається з даними, коли їх переводять у рівень Archive, і які обмеження це створює?
10. Що таке «soft delete» у Azure Blob Storage і як воно захищає дані від випадкового видалення?

Лабораторна робота №2.

Синхронізація локального сервера з Azure File Share та основи роботи з Azure File Share в Python за допомогою Azure SDK

Завдання

Частина 1. Набути навичок створення рішення для синхронізації файлових серверів із одним спільним Azure File Share.

Частина 2. Навчитися створювати та керувати Azure File Share за допомогою класів бібліотеки azure.storage.fileshare.

Теоретичні відомості

Azure File Shares — це служба зберігання файлів в хмарі від Microsoft Azure, яка дозволяє створювати загальні файлові сховища для зберігання та обміну файлами в хмарному середовищі. Ця служба працює за протоколом SMB (Server Message Block), що дозволяє користувачам легко мапити (монтувати) ці файлові сховища на свої локальні пристрої, подібно до локальних мережевих дисків.

Основні характеристики Azure File Shares:

Спільний доступ до файлів: Azure File Shares надає можливість організаціям ділитися файлами між різними серверами та користувачами через мережу.

Протоколи: Використовує SMB та NFS (для Linux середовищ) для доступу до файлів.

Інтеграція: Легко інтегрується з Windows, Linux і macOS системами. Можна мапити файлові ресурси через стандартні засоби операційних систем.

Доступність: Azure File Shares забезпечує високу надійність та доступність, тому файли доступні з будь-якої точки світу через Інтернет.

Резервне копіювання: Azure забезпечує механізми для резервного копіювання файлів і відновлення їх у разі втрати.

Шифрування: Всі дані в Azure File Shares шифруються як в стані спокою, так і під час передачі.

Сценарії використання: Спільне використання файлів між серверами.

Зберігання та обмін файлами для мобільних і віддалених користувачів.

Інтеграція із застосунками, що потребують файлових сховищ.

Хід роботи – Частина 1

1. Зайдіть на портал Azure - https://portal.azure.com
2. Створіть групу ресурсів, яка включатиме: віртуальну мережу №1 (назва мережі має містити ваше прізвище), підмережу №1 (subnet) з віртуальною машиною №1. (Доданок 1)
3. Створіть обліковий запис сховища Azure (назва сховища має містити ваше прізвище). Завантажте файли у File Shares.

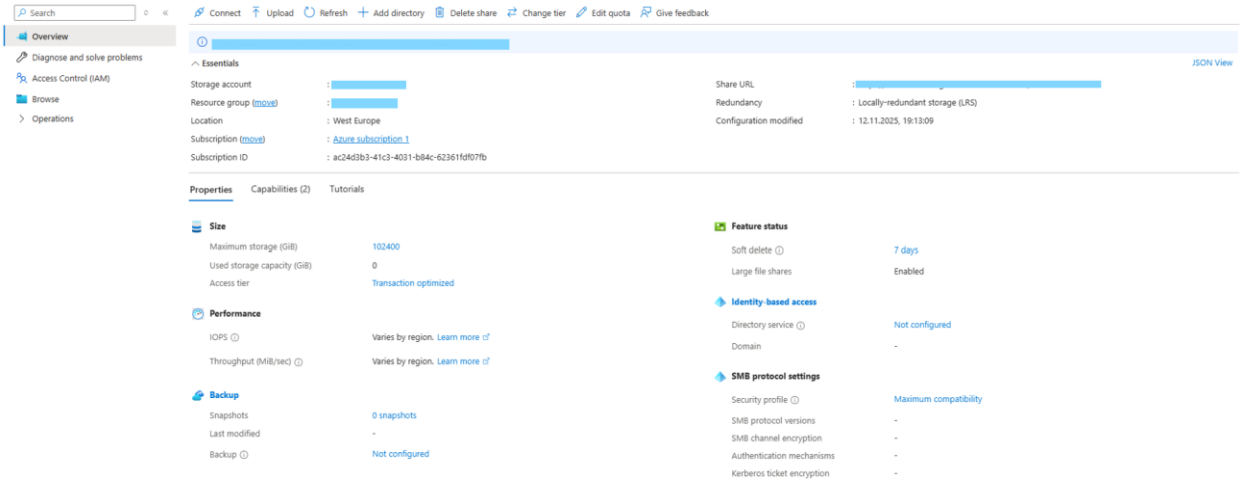


Рис. 1. Сховище даних типу File Shares

4. Створіть Snapshots (знімки) для SMB File Shares.

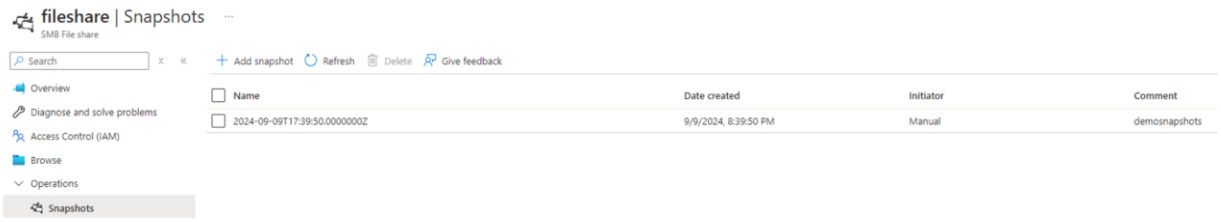


Рис. 2. Snapshots (знімки) для SMB File Shares

5. Отримайте ключ для встановлення зв’язку між папками з файлами на локальному диску та в хмарі (ключ можна завантажити за посиланням: Storage → Data Share → FileShare → Connect).

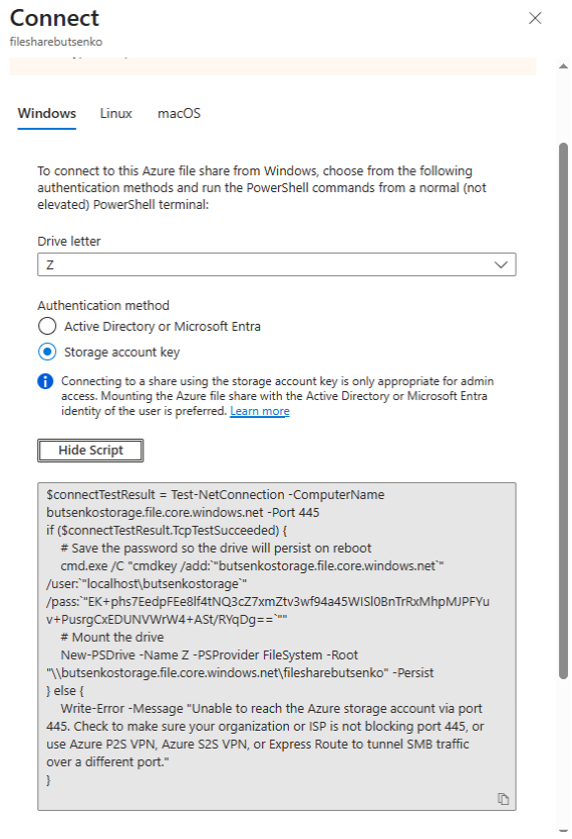


Рис. 3. Генерація ключа для встановлення зв’язку між папками з файлами на локальному диску та в хмарі

6. Перейдіть на ВМ та встановіть ключ в оболонці PowerShell.

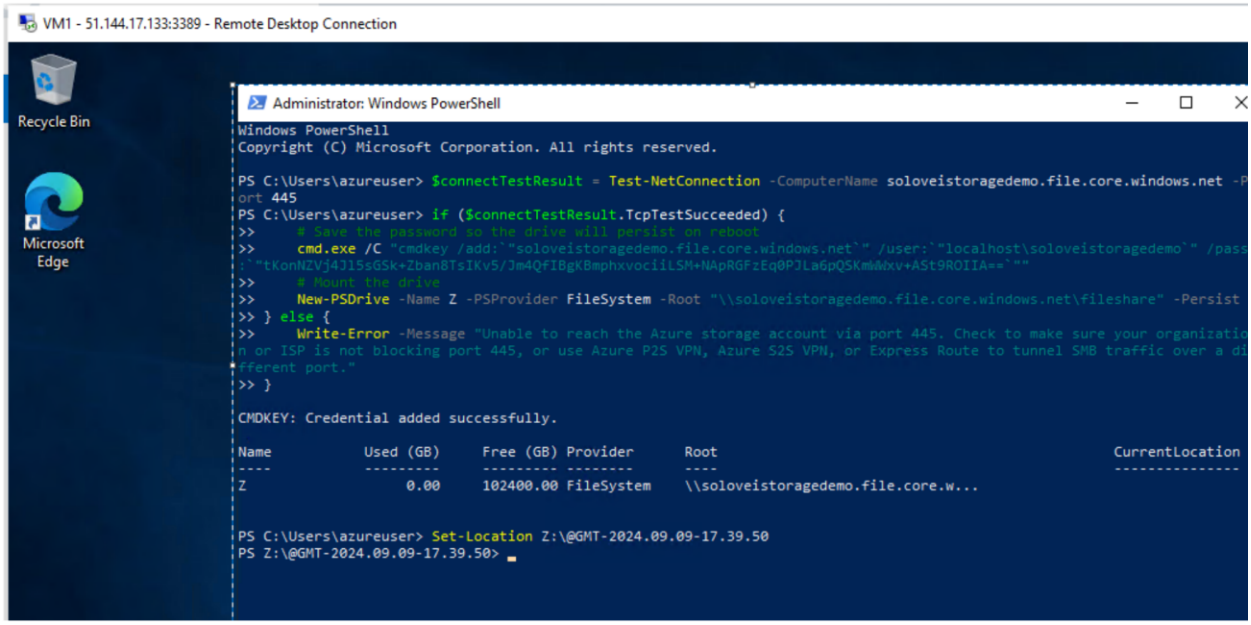


Рис. 4. Інсталяція ключа на віртуальній машині

7. Створіть файл на ВМ і завантажте його у приєднану папку з файлами.

8. Перевірте, що файли на локальному диску вашої ВМ синхронізовані з файлами у сховищі Azure.

Хід роботи – Частина 2

1. Створіть об'єкт ShareServiceClient через connection string. Ініціалізуйте об'єкт share_client.

2. Створіть File Share з назвою [ваше_прізвище]labshare. Якщо такий вже існує — обробіть виняток ResourceExistsError.

3. Створіть директорію [ваше_прізвище]labfolder у межах File Share. Якщо така директорія вже існує — обробіть це відповідно.

4. Завантажте 5 локальних файлів у директорію [ваше_прізвище]labfolder в Azure.

5. Отримайте список усіх файлів та папок у директорії [ваше_прізвище]labfolder. Виведіть їхні імена в консоль.

6. Видаліть усі файли з хмарної директорії. Якщо файл відсутній — обробіть ситуацію.

Підготуйте звіт, який включає: знімки екрана з результатами кроків 1-8 (частина 1) та кроку кроків 4-5 (частина 2); лістінг програми; відповіді на контрольні запитання.

Контрольні запитання

1. Що таке Azure File Share і які протоколи він підтримує для доступу до файлів?

2. У чому відмінність між SMB та NFS при використанні File Share?

3. Які типи сховищ підтримують знімки (snapshots)? Для чого вони використовуються?

4. Як створити та підключити віртуальну машину до File Share через PowerShell?

5. Що відбувається при створенні файлу у змонтованому File Share на віртуальній машині?

6. Для чого використовується клас ShareServiceClient у бібліотеці azure.storage.fileshare?

7. Яким чином можна отримати список усіх файлів у хмарній директорії?

Доданок 1

```
$grp="VMdemo"
$location="westeurope"
$vnetName1="VNET1"
$subnetName1="SUBNET_1"
$vmName1 = "VM1"
# CREATE RESOURCE GROUP
az group create --name $grp --location $location

# CREATE VIRTUAL NETWORK
az network vnet create --address-prefixes 10.0.0.0/16 --name $vnetName1 --resource-group $grp
# CREATING SUBNET
az network vnet subnet create -g $grp --vnet-name $vnetName1 -n $subnetName1 --address-prefixes 10.0.0.0/24
# CREATING VM1
az vm create --resource-group $grp --name $vmName1 --image MicrosoftWindowsServer:WindowsServer:2019-Datacenter:latest --vnet-name $vnetName1 --subnet $subnetName1 --admin-username XXXX --admin-password XXXX --size Standard_B2s
```

Лабораторна робота №3

Синхронізація локального сервера з Azure File Share за допомогою компонента Azure File Synch

Мета роботи: Здобути навички створення рішення синхронізації файлових серверів з одним спільним Azure File Share.

Завдання

Створити рішення синхронізації файлових серверів з одним спільним Azure File Share.

Теоретичні відомості

Azure File Sync — сервіс, що дозволяє синхронізувати локальні файлові сервери з хмарним сховищем File Share. Основні компоненти:

- ✓ Storage Sync Service — ресурс у Azure, що керує синхронізацією.
- ✓ Sync Group — група синхронізації (включає один Azure File Share і один або більше серверів).
- ✓ Azure File Sync Agent — компонент, який встановлюється на локальний або хмарний сервер (ВМ).

Azure File Sync дозволяє: робити зеркальну синхронізацію файлів, використовувати Azure як резервне сховище, зменшувати локальне навантаження через cloud tiering (не використовується у даній лабораторній)

На віртуальну машину Windows Server потрібно завантажити інсталяційний пакет з офіційного сайту Microsoft. Після встановлення виконується: Registration — реєстрація сервера в Storage Sync Service. Сервер з'являється в розділі Registered Servers. Це дозволяє Azure контролювати синхронізацію файлів.

Для створення процедури синхронізації (Sync Group) потрібно створити: Sync Group, Вказати Azure File Share (хмарна сторона), Призначити Server Endpoint — локальний диск і папку на ВМ. Server Endpoint визначає: шлях до папки (наприклад, D:\SyncFolder), політику кешування, параметри синхронізації. Після цього файли автоматично передаються між локальним диском і хмарним сховищем.

Перевірку синхронізації можна виконати двома способами: 1) додати файл на локальний диск → він з’явиться в File Share; 2) завантажити файл у File Share → він з’явиться на ВМ.

Синхронізація відбувається майже миттєво, залежно від навантаження.

Хід роботи

1. Зайдіть на портал Azure - https://portal.azure.com

2. Створіть групу ресурсів, яка включатиме: віртуальну мережу №1 (назва мережі має включати ваше прізвище), підмережу №1 (sub-net) з віртуальною машиною №1;

3. Створіть обліковий запис сховища Azure. Завантажте файли в File Shares.

4. Додайте диск до ВМ в Azure Resource Manager.

5. Зайдіть на віртуальну машину - доданий диск має статус «Unallocated». Виконайте «New simple Volume» – визначте літеру, і відформатуйте диск (рис. 1). Перевірте, що новий диск додано до ВМ.

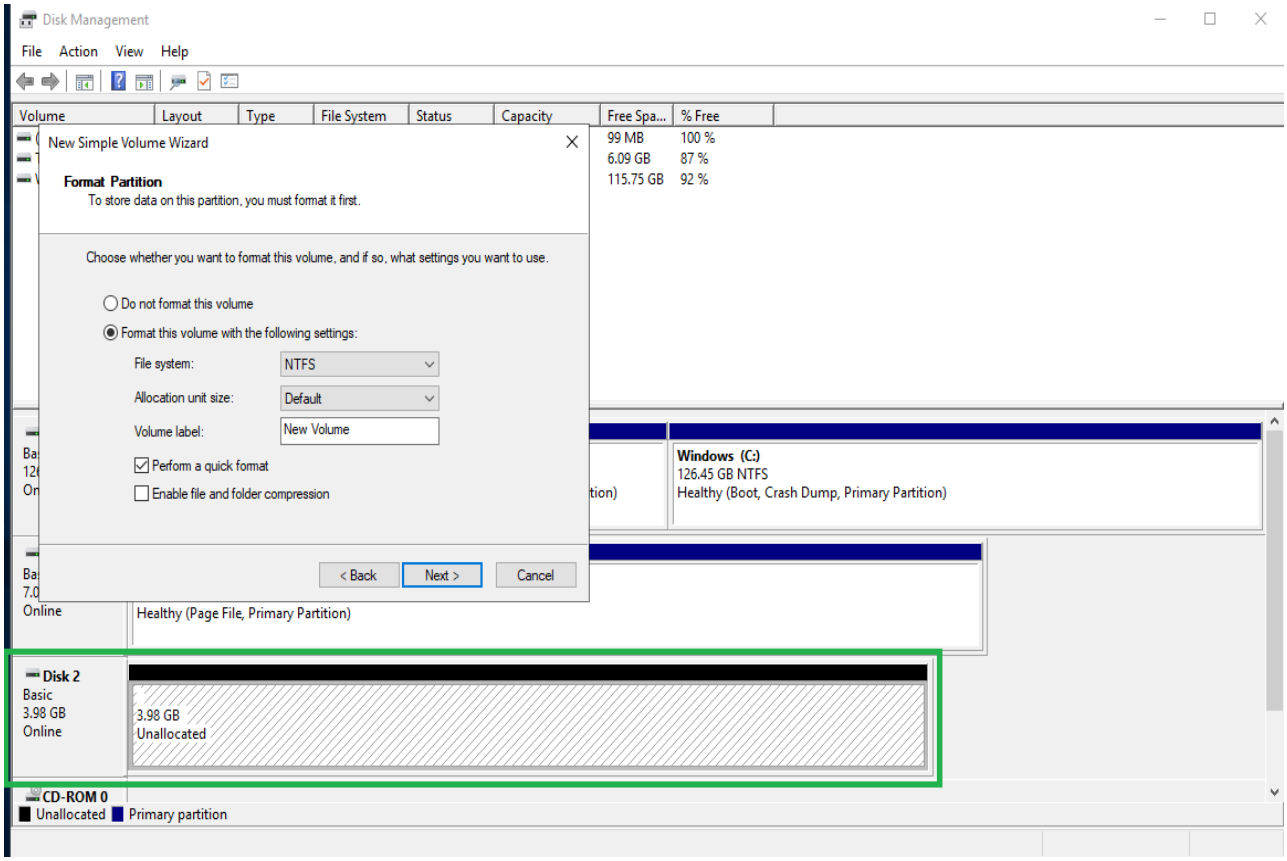


Рис. 1. Форматування доданого диску на ВМ

6. На ВМ – відключить IE Enhanced Security Configuration, використовуйте Server Manager (рис. 2).

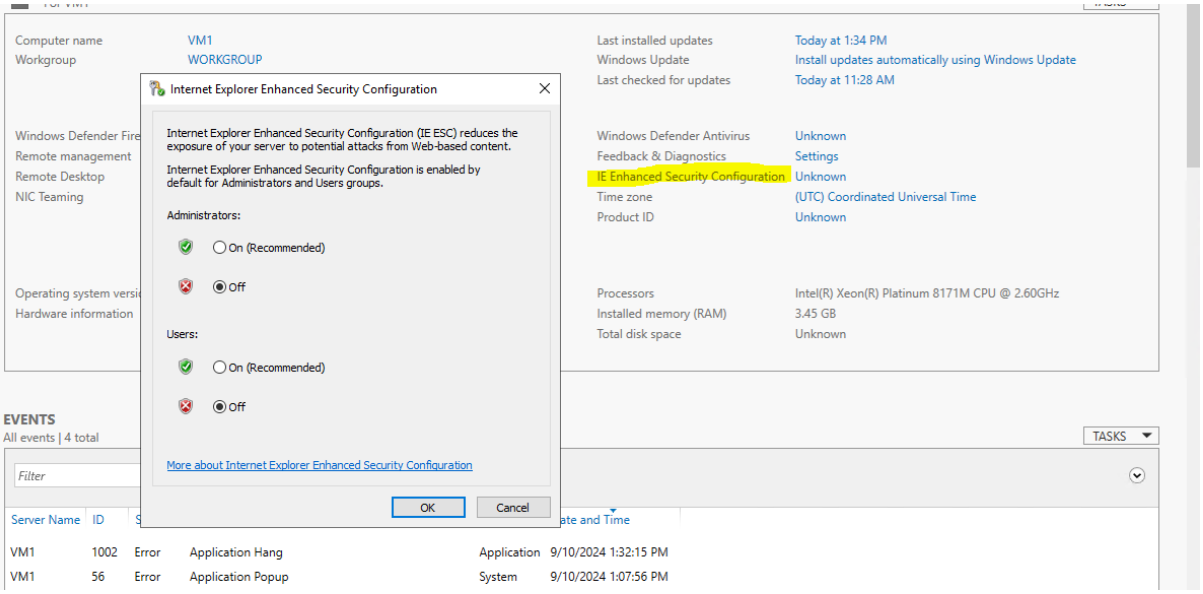


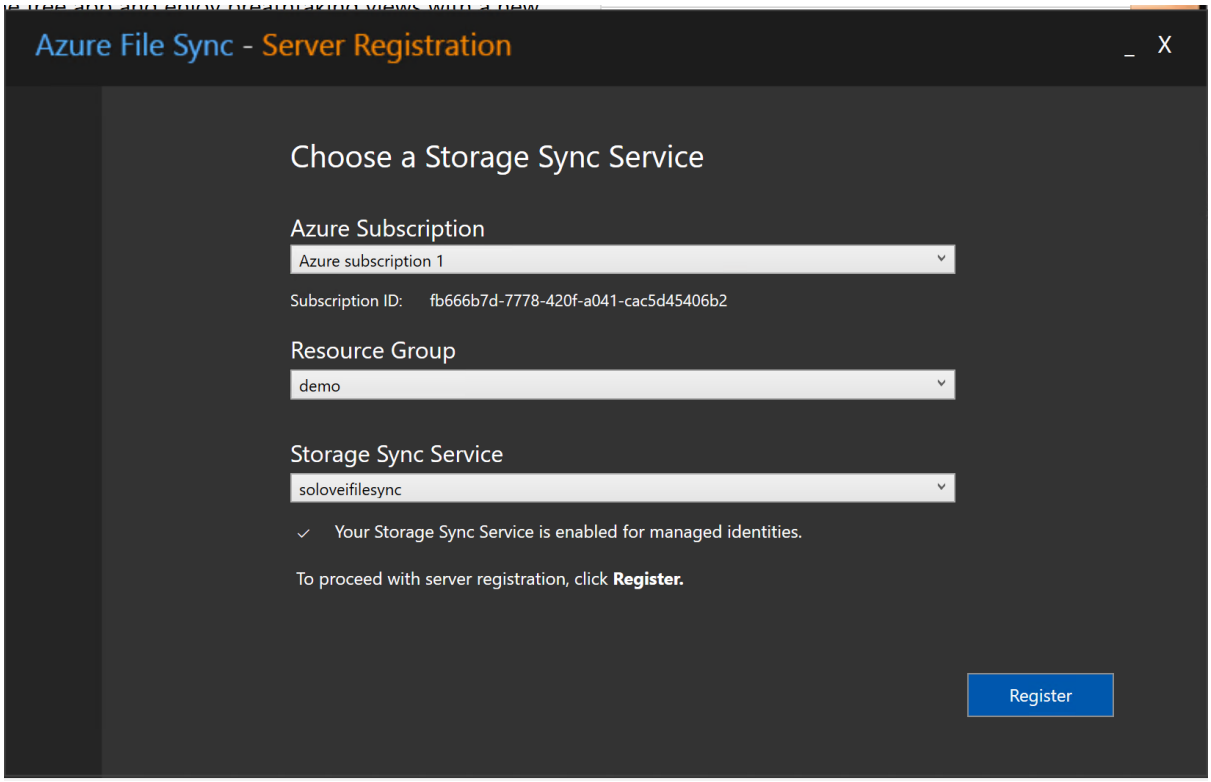
Рис. 2. Відключення IE Enhanced Security Configuration на ВМ

7. В Azure Resource Manager, створіть ресурс Azure File Synch.

8. Завантажте з інтернету компонент для синхронізації (Azure File Synch Agent). Download Azure File Sync Agent from Official Microsoft Download Center

9. Виконайте завантаження та встановлення компонент Azure File Synch Agent на ВМ.

10. Розгорніть компонент Azure File Synch Agent на ВМ.



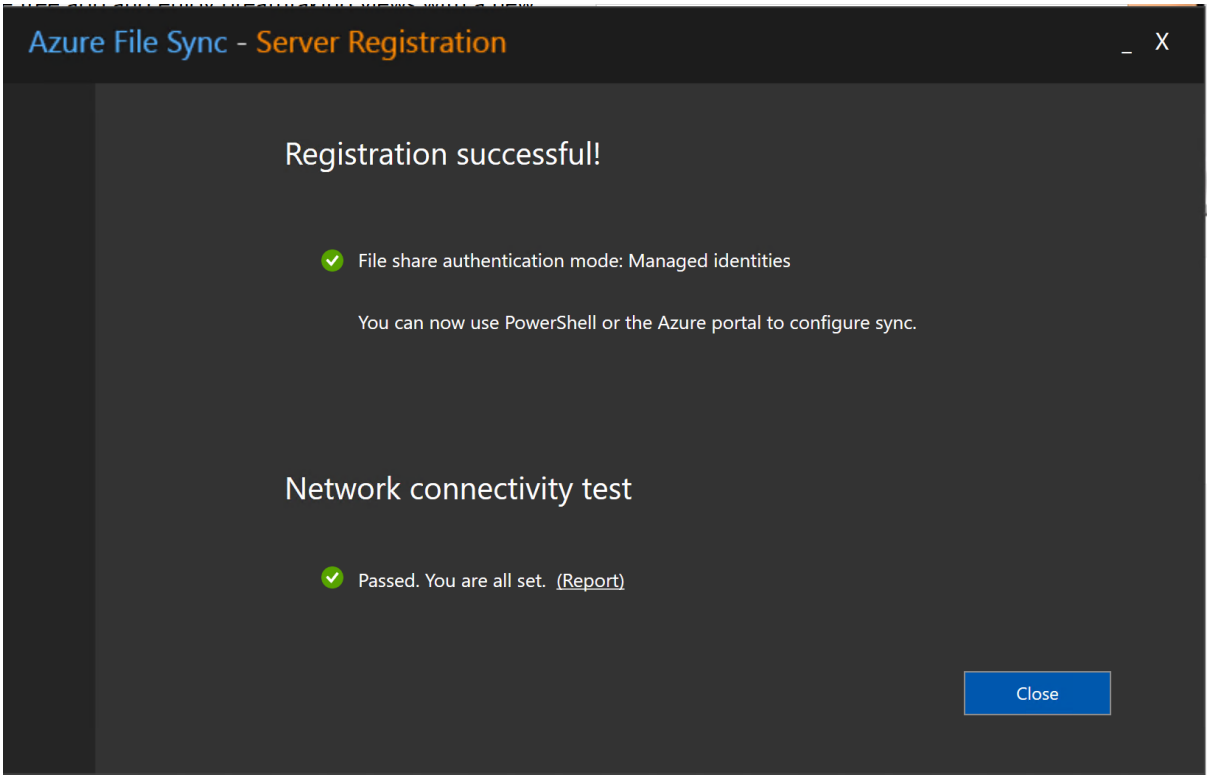


Рис. 3. Розгортання компоненту Azure File Synch Agent на ВМ

11. Зареєструйте сервер, перевірте, що сервер з ім'ям вашої ВМ додано до ресурсу Azure File Synch/Registered servers (рис. 4).

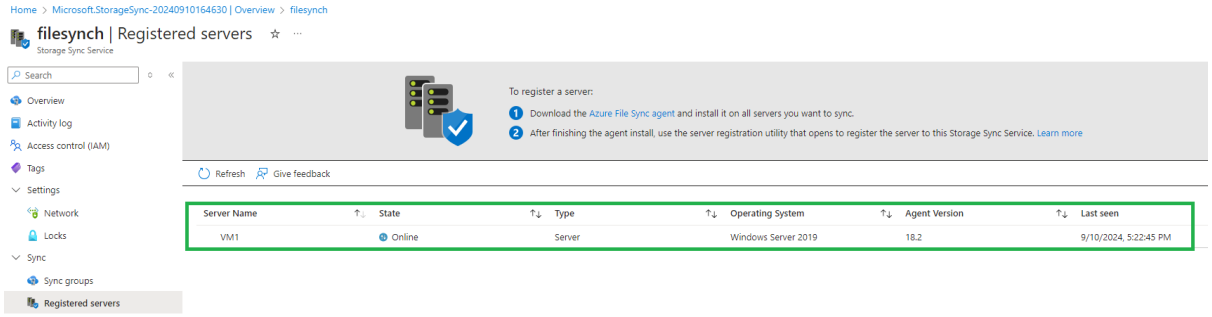


Рис. 4. Зареєстрований сервер додано до компонента Azure File Synch Agent в Azure Resource Manager

12. В Azure платформі створіть процедуру синхронізації, визначте диск на вашій ВМ, дані на якому будуть синхронізовано з Azure File Share. Зв'яжіть локальний сервер зі сховищем Azure (рис. 5).

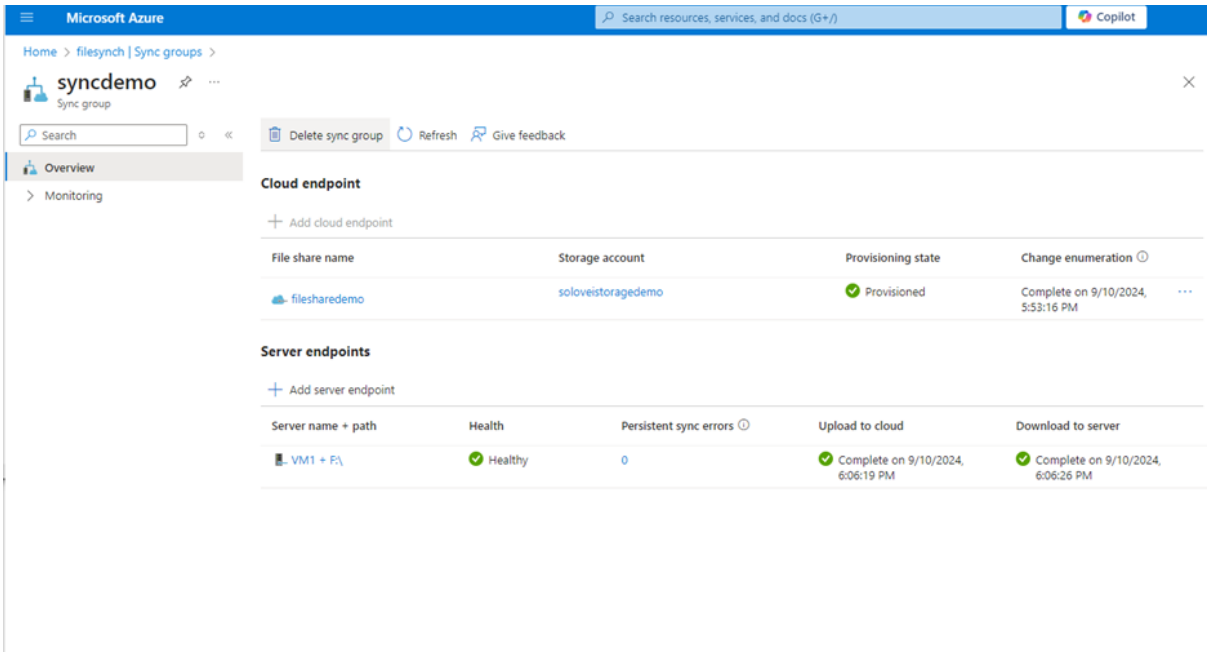


Рис. 5. Створення процедури синхронізації.

13. Перевірте, що файли на локальному диску вашій ВМ синхронізовані з файлами в сховище Azure.

14. Підготуйте звіт, який включає: знімки екрана з результатами кроків 11-13; відповіді на контрольні запитання.

15. Видаліть всі створені ресурси – у наступні послідовності – сервер синхронізації, процедуру синхронізації, всі інші ресурси.

Контрольні запитання

1. Чим відрізняються snapshots від повноцінних резервних копій?
2. Як працює доступ до попередніх версій файлів (Previous Versions) у SMB-шерінгу?
3. Які переваги підходу “Lift and Shift” під час міграції файлових серверів у хмару?
4. У яких випадках Lift and Shift-міграція не рекомендується і чому?
5. Які обмеження існують у Azure File Share для SMB-протоколу?
6. Як Python SDK дозволяє працювати з Azure File Share: створення файлів, папок і snapshot’ів?
7. Які відмінності між File Share та Blob Storage в Azure для сценаріїв з SMB?
8. Як відбувається синхронізація серверів за допомогою Azure File Sync Agent?
9. Що таке Cloud Tiering у Azure File Sync і як воно працює?
10. Як забезпечується консистентність даних між локальним сервером і Azure File Share?
11. Які метрики важливо моніторити у File Sync-інфраструктурі (latency, sync status, journal size тощо)?

Лабораторна робота №4

Автоматизовані процеси для роботи з повідомленнями з Azure Queue Storage

Мета роботи: Розробити автоматизований процес, який буде періодично перевіряти Azure Queue, зчитувати повідомлення, зберігати їх у Azure Blob Storage, а потім видаляти повідомлення з черги.

Теоретичні відомості

Azure Table Storage — це нереляційне NoSQL-сховище, яке підтримує збереження великих обсягів структурованих даних у вигляді таблиць. Кожен запис у таблиці називається Entity (сутність). Сутність не має фіксованої схеми, тому вона може містити довільну кількість полів, що робить Table Storage гнучким та ефективним для розподілених систем. Обов'язковими елементами сутності є PartitionKey, RowKey і Timestamp.

PartitionKey визначає логічний розподіл даних між партиціями. Усі сутності з однаковим PartitionKey фізично зберігаються разом, що забезпечує швидкий доступ до групи пов'язаних даних. Від правильного вибору PartitionKey залежить масштабованість та продуктивність запитів. RowKey слугує унікальним ідентифікатором сутності всередині конкретної партиції. Разом PartitionKey + RowKey формують унікальний ключ запису, аналог первинного ключа в реляційних БД. Timestamp — системна властивість, яка автоматично оновлюється при зміні сутності та використовується для механізмів реплікації. Інші поля сутності називаються Properties; вони можуть містити різні типи даних і формують гнучку структуру таблиці.

Azure Table Storage підтримує просту й ефективну модель доступу до даних. Найшвидшими є точкові запити (Point Queries), які використовують повний унікальний ключ: PartitionKey та RowKey. Такий запит завжди повертає не більше однієї сутності та виконується з мінімальною затримкою.

Діапазонні запити (Range Queries) дозволяють отримати всі сутності в межах одного PartitionKey, але з фільтром по RowKey, наприклад RowKey > "100" AND RowKey < "500". Вони широко застосовуються для вибірок за часом, індексованими значеннями чи зберіганням подій у порядку. Table Storage також підтримує фільтрацію за Properties, однак найефективнішими є саме фільтри, що накладаються на PartitionKey або RowKey, оскільки вони

використовують індексацію. Фільтрація за іншими полями може вимагати сканування частини таблиці. Сортування в Table Storage обмежене: дані всередині партиції автоматично впорядковуються за RowKey, тому користувач може отримувати результати у вже відсортованому вигляді, але додаткове серверне сортування не підтримується.

Azure Table Storage ефективно використовується у сценаріях, де потрібне високопродуктивне масштабоване сховище для великих обсягів структурованих, але не реляційних даних. Завдяки гнучкій схемі зберігання він добре підходить для телеметрії, логування подій, збереження станів обробки даних, конфігурацій та довідників. Розподіл даних за PartitionKey дозволяє підтримувати високі навантаження та горизонтальне масштабування.

Сервіс часто використовується в архітектурах з мікросервісами, де кожен сервіс може зберігати власні сутності у таблиці зі специфічною структурою. Table Storage легко інтегрується з іншими сервісами Azure, такими як Azure Functions, Logic Apps, Event Grid. Наприклад, функція може обробляти вхідні дані й записувати результат у Table Storage, або навпаки — реагувати на зміни у таблиці. Завдяки низькій вартості та високій швидкості операцій Table Storage є оптимальним рішенням для систем моніторингу, IoT-платформ, журналювання та зберігання великої кількості дрібних структурованих записів**.**

Завдання

Частина 1. Робота з Logic App.

Частина 2. Програмна реалізація на Python.

Хід роботи

1. Зайдіть на портал Azure – https://portal.azure.com
2. Створіть Azure Logic App з тригером "When there are messages in the queue" (поява повідомлення в черзі).
3. Налаштуйте Logic App для зчитування повідомлення з черги.
4. Додайте дію, яка зберігає вміст повідомлення у новий файл у Azure Blob Storage.

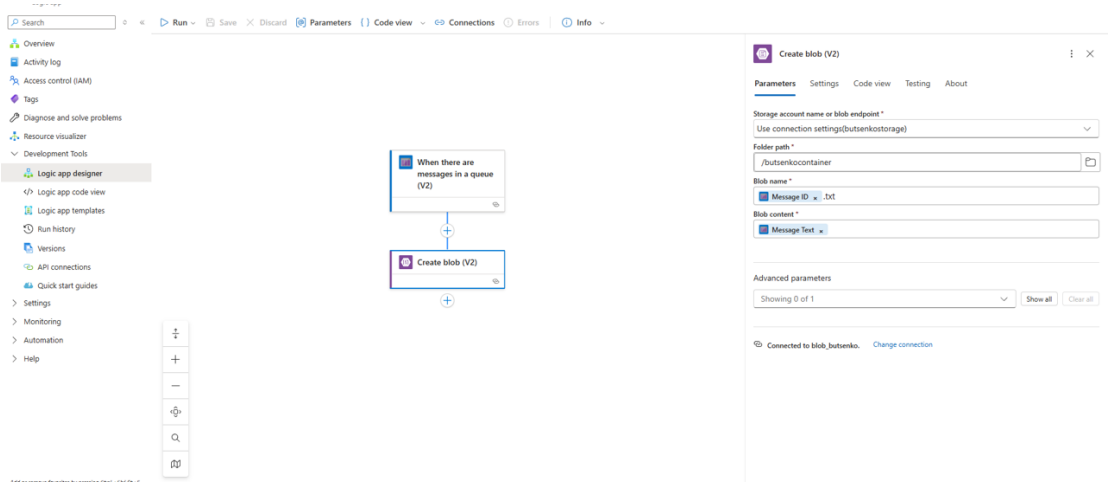


Рис. 1. Процес в Logic App, для отримання повідомлень з черги та збереження у Azure Blob Storage

5. Налаштуйте Postman (або інший інструмент) для надсилання тестового повідомлення до черги.

6. Перевірте, що повідомлення із черги зберігається у Blob Storage та видаляється після обробки.

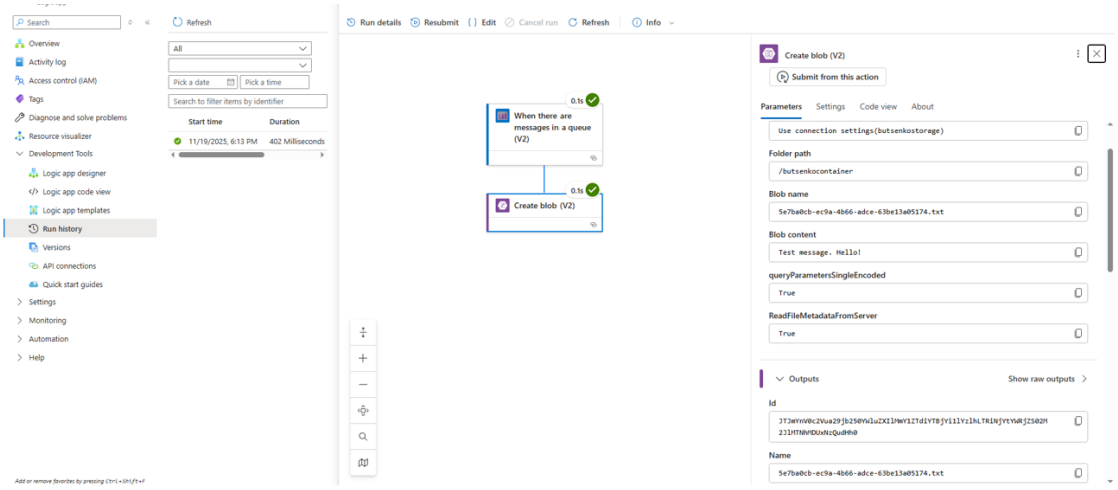


Рис. 2. Успішно виконаний процес в Logic App

7. Напишіть програму на Python, яка:

a. Підключається до Azure Storage за допомогою рядка підключення (Connection String).

b. Кожні 10 секунд перевіряє чергу myqueue на наявність нових повідомлень.

c. Якщо повідомлення знайдено:

d. Зчитує текст повідомлення.

e. Створює текстовий файл з унікальним ім’ям (наприклад, message-<uuid>.txt) та завантажує його у контейнер container2messages.

f. Видаляє повідомлення з черги після обробки.

g. Якщо повідомлень немає — просто очікує 10 секунд і перевіряє знову.

8. Підготуйте звіт, який включає: знімки екрана; програму; відповіді на контрольні запитання.

9. Видаліть всі створені ресурси – у наступні послідовності – сервер синхронізації, процедуру синхронізації, всі інші ресурси.

Контрольні запитання

1. Які основні характеристики Azure Queue (розмір повідомлення, кількість черг, типи черг)?
2. Яку структуру має Message у Queue Storage та які обмеження накладаються на його вміст і розмір?
3. Як працює механізм visibility timeout після отримання повідомлення з черги?
4. У чому різниця між Peek та Dequeue операціями в Azure Queue Storage?
5. Які переваги використання Azure Logic Apps для обробки повідомлень з Azure Queue Storage?
6. Який тригер використовується в Logic Apps для автоматичного зчитування нових повідомлень з черги?
7. Як можна реалізувати обробку повідомлень у циклі в Logic Apps при великому потоці даних?
8. Що відбувається з повідомленням після того, як Logic App успішно його опрацює?
9. Як налаштувати error handling у Logic Apps при виникненні помилки під час обробки повідомлення з черги?
10. Яку бібліотеку Python використовується для роботи з Azure Queue Storage, та як її встановити?
11. Як створити клієнт для черги за допомогою QueueServiceClient або QueueClient?

Лабораторна робота №5
Оптимізація доступу до даних у Azure Table Storage

Мета роботи: Ознайомитись із принципами побудови ефективної структури таблиць у Azure Table Storage, протестувати три різні стратегії зберігання, виконати запити та проаналізувати їх продуктивність.

Теоретичні відомості

Azure Table Storage — це NoSQL-сховище типу key-value, оптимізоване для масштабованих і високопродуктивних операцій читання та запису. Таблиці в Azure не мають фіксованої схеми: кожен запис (entity) може містити свій набір властивостей. Ключові елементи структури:

PartitionKey — визначає логічний розподіл даних на партиції. Дані з різними PartitionKey можуть розміщуватися на різних фізичних вузлах.

RowKey — унікальний ключ всередині партиції.

PartitionKey + RowKey — унікальний складений первинний ключ сутності.

Timestamp — системне поле, яке Azure оновлює автоматично.

Від структури ключів залежить продуктивність запитів, масштабованість та вартість операцій.

Оскільки Azure Table Storage є масштабованим NoSQL-сховищем, від правильної структури ключів залежить швидкість операцій.

У лабораторній використовуються три популярні стратегії:

1. Стратегія оптимізована для читання оптимізує: швидкі точкові запити (Point queries), пошук конкретного замовлення клієнта. Переваги: Мінімальна затримка при вибірці одного запису. Логічне групування даних за клієнтом спрощує звітність. Недоліки: Неможливо ефективно фільтрувати за хронологією без додаткових індексів.

2. Стратегія "складені ключі для діапазонного читання" дозволяє: робити ефективні range queries (по датах), виконувати сортування по RowKey (лексикографічне). Приклад: Azure Table Storage зберігає рядки в порядку RowKey, тому комбінований ключ YYYYMMDD_orderID дозволяє отримувати всі замовлення клієнта за конкретний місяць або період, уникати повного сканування таблиці.

3. Стратегія "Entity Duplication (Дублювання сутностей)" створює дві копії одних і тих самих даних, це класичний NoSQL-підхід, коли один і той самий документ зберігається у двох видах, щоб оптимізувати доступ із різних напрямків. Переваги: Швидкі запити і за order_id, і за customer_id. Уникнення дорогих операцій сканування. Недоліки: Дані дублюються → потрібен контроль consistency при оновленнях.

Завдання

1. Перейдіть на платформу Kaggle Datasets. Завантажте будь-який CSV-файл, що містить дані про транзакції, замовлення або клієнтів. Наприклад: E-commerce Data, Online Retail. Оберіть не менше 1000 записів для тестування.
2. Оберіть з файлу наступні поля (або подібні до них): customer_id, order_id або transaction_id, order_date, amount або total_price.
3. Створіть три таблиці з різними стратегіями зберігання

3.1. для ефективного читання (Read-efficient design) - використовуйте PartitionKey = customer_id, RowKey = order_id.

3.2. Compound Keys – використовуйте

PartitionKey = customer_id

RowKey = order_date_order_id (наприклад: 20230901_order001)

3.3. Entity Duplication – використовуйте

PartitionKey = customer_id, RowKey = order_id

PartitionKey = order_id, RowKey = customer_id

4. Для кожної таблиці виконайте:

4.1.Точковий запит (Point query):

PartitionKey eq 'customer_1' and RowKey eq 'order_123'

4.2.Діапазонний запит (Range query):

PartitionKey eq 'customer_1' and RowKey ge '20230901' and RowKey le '20230930' (для Compound Keys)

5. Заповніть таблицю

Таблиця	**Тип запиту**	**Кількість результатів**	**Час виконання (сек)**
Read-Efficient	Point		
Read-Efficient	Range		
Compound Keys	Range		
Entity Duplication	Point (order)		
Entity Duplication	Point (user)		

6. Підготуйте звіт, який включає: CSV-файл, який використовувався. Код на Python з імпортом, записом і запитами. Таблицю з результатами вимірювань часу. Відповіді на контрольні запитання.

7. Видаліть всі створені ресурси.

Контрольні запитання

1. Яка стратегія забезпечила найшвидший точковий запит? Чому?
2. Для чого використовуються складені ключі? Яку перевагу вони дають?
3. Які переваги і недоліки дублювання сутностей?
4. Чому в Azure Table Storage важливо правильно вибирати PartitionKey?
5. Які обмеження має Table Storage при фільтрації даних?

Лабораторна робота №6

Робота з Azure Cosmos DB for NoSQL

Мета роботи: Здобути навики роботи з Azure Cosmos DB за допомогою мовизапитів SQL і клієнтських бібліотек для .NET, JavaScript, Python і Java.

Теоретичні відомості

Azure Cosmos DB підтримує п’ять рівнів узгодженості, що дозволяють балансувати між точністю даних, продуктивністю та затримкою: Strong, Bounded Staleness, Session, Consistent Prefix та Eventual. Вибір рівня узгодженості визначає, наскільки актуальними будуть дані, які читає клієнт після запису, особливо в глобально розподілених системах.

Strong Consistency гарантує найвищий рівень — усі читання відбуваються з повною синхронізацією даних у всіх репліках, що забезпечує поведінку, аналогічну ACID-системам. Однак цей режим збільшує затримку та знижує доступність у мульти-регіональних конфігураціях. Bounded Staleness дозволяє читачеві бачити дані з контрольованою затримкою (за часом або кількістю операцій), забезпечуючи баланс між точністю та продуктивністю. Session Consistency є режимом за замовчуванням і забезпечує семантику read-your-own-writes у межах окремої клієнтської сесії, що робить його оптимальним для більшості бізнес-сценаріїв. Consistent Prefix гарантує правильну послідовність записів, але не їх актуальність. Eventual Consistency забезпечує максимальну продуктивність і мінімальну затримку, але читання може тимчасово повертати застарілі дані, що прийнятно для телеметрії, логів та інших нечутливих до негайної точності систем.

Cosmos DB пропонує два основні режими керування ресурсами, що визначають, як споживаються й оплачуються обчислювальні ресурси: Provisioned Throughput Mode та Serverless Mode.

У Provisioned Throughput Mode користувач задає наперед кількість Request Units (RU/s), доступних для контейнера або бази даних. Це забезпечує гарантовану продуктивність і передбачувану затримку. Такий режим оптимальний для стабільних або високонавантажених систем, наприклад онлайн-магазинів, фінансових сервісів або мікросервісів із постійним потоком запитів. Недоліками є потенційна переплата в періоди низького навантаження та необхідність планування потрібного RU-ліміту.

Serverless Mode — це споживацька модель оплати, за якої ресурси не резервуються наперед, а оплата відбувається лише за фактичні RU, використані запитами. Режим ідеально підходить для нерегулярних, непередбачуваних або низькоактивних робочих навантажень, таких як невеликі веб-сервіси, прототипи, тестові середовища, серверless-архітектури. Проте при високих і стабільних навантаженнях він може бути значно дорожчим порівняно з Provisioned Throughput.

Cosmos DB використовує механізм секціонування для масштабування як обсягу даних, так і пропускної здатності. Логічна секція (logical partition) — це група документів, що мають однакове значення Partition Key. Логічна секція визначає логічне розбиття даних та є одиницею транзакції в межах ACID-операцій. Дані всередині логічної секції можуть взаємодіяти у межах єдиної транзакції, тоді як крос-секційні транзакції обмежені.

Фізична секція (physical partition) — це фактичний блок зберігання та обчислювальних ресурсів, на які Cosmos DB розподіляє логічні секції. Одна фізична секція може містити багато логічних секцій. Система автоматично створює нові фізичні секції при зростанні обсягу даних або навантаження, що забезпечує горизонтальне масштабування без втручання користувача. Вибір ефективного Partition Key є критичним і впливає на рівномірність навантаження, уникнення гарячих секцій (hot partitions) і оптимальне використання RU. Добре підібраний ключ повинен мати високу кардинальність, рівномірний розподіл та враховувати тип основних запитів.

Завдання

1. Розгорнути Cosmos DB Emulator (https://learn.microsoft.com/en-us/azure/cosmos-db/emulator).

2. Завантажити JSON file за посиланням (https://jsonplaceholder.typicode.com/users)

3. За допомогою однієї з клієнтських бібліотек:

a. створити базу даних Cosmos DB та контейнер.

b. завантажити JSON документ; визначте ключ контейнера.

c. виконати SQL запити:

- ✓ читання з фільтром.
- ✓ оновлення значення властивості.

- ✓ Визначити кількість ОЗ/с, які знадобились для кожного запиту.

4. Підготуйте звіт, який включає: програму. Таблицю з результатами ОЗ/с, які знадобились для кожного запиту. Відповіді на контрольні запитання.

Контрольні запитання

1. Чому рівень Strong Consistency забезпечує найвищу узгодженість, але впливає на продуктивність і глобальну доступність?
2. У яких сценаріях доцільно використовувати Eventual Consistency, і які ризики це створює для читання даних?
3. Що означає властивість read-your-own-writes і для яких рівнів узгодженості вона гарантується?
4. У чому полягає суть режиму Provisioned Throughput Mode, і як вимірюється пропускна здатність Cosmos DB?
5. Які переваги та недоліки використання заздалегідь виділеного пропускного режиму (Provisioned Throughput) для стабільних навантажень?
6. Як працює Serverless Mode, і в яких випадках він є оптимальним вибором?
7. Чому Serverless Mode може бути дорожчим для високих постійних навантажень порівняно з Provisioned Throughput?
8. Що таке RU (Request Units) і як їх споживання відрізняється між цими двома режимами?
9. Що таке логічна секція (logical partition) і яку роль відіграє Partition Key?
10. У чому полягає відмінність між логічною та фізичною секцією (physical partition)?
11. Яким чином Cosmos DB автоматично масштабує фізичні секції при збільшенні обсягу даних або навантаження?
12. Які критерії слід враховувати під час вибору ефективного Partition Key?
13. Які основні об’єкти становлять ієрархію SDK azure.cosmos (наприклад, CosmosClient, Database, Container) і яку роль виконує кожен із них?

Лабораторна робота №7

Підключення IoT-пристрою до Azure IoT Hub та збереження телеметрії у InfluxDB Cloud

Мета роботи: Ознайомитись із принципами роботи хмарних сервісів IoT Hub (Microsoft Azure) та InfluxDB Cloud.

Теоретичні відомості

IoT Hub — це керований хмарний сервіс Azure, який забезпечує: двосторонній обмін даними між IoT-пристроями та хмарою, масштабовану обробку телеметрії, автентифікацію пристроїв, збирання, маршрутизацію та зберігання повідомлень.

Основні елементи IoT Hub:

- ✓ Device Identity Registry — реєстр, де зберігаються всі зареєстровані пристрої.
- ✓ Shared Access Keys — ключі для автентифікації.
- ✓ Built-in Event Hub-compatible endpoint — точка доступу для читання телеметрії сторонніми сервісами.
- ✓ Routes — правила пересилання повідомлень у різні сервіси Azure.

Кожен пристрій в IoT Hub має: Device ID — унікальне ім’я. Authentication keys — Primary / Secondary Key.

Connection String — рядок підключення, який використовується у пристрої.

Формат Connection String:

HostName=<hub>.azure-devices.net;DeviceId=<device-id>;SharedAccessKey=<key>

Веб-симулятор Raspberry Pi працює як віртуальний пристрій, який надсилає дані сенсорів (температуру, вологість, світло тощо) у IoT Hub, використовує Node.js-подібний код для відправки повідомлень. Після вставлення Connection String, пристрій імітує роботу реального сенсорного вузла.

IoT Hub містить внутрішній Event Hub endpoint, який дозволяє читати дані телеметрії з IoT Hub так само, як із Event Hubs.

Параметри: Event Hub-compatible endpoint — URL для підключення клієнта. Event Hub-compatible name — логічна назва потоку. Consumer Group — група читачів (за замовчуванням $Default).

Читання телеметрії здійснюється через бібліотеку azure-eventhub.

InfluxDB — це високошвидкісна time-series база даних, оптимізована для: телеметрії, потокових IoT-даних,

вимірювань сенсорів.

Основні компоненти InfluxDB: Bucket — логічне сховище (аналог БД або таблиці). Organization — організаційна область користувача. Token — ключ для аутентифікації API. Measurement — таблиця, що містить time-series записи. Fields — значення (temperature, humidity). Tags — індексовані метадані (наприклад, device_id).

У InfluxDB кожен запис має:

- ✓ автоматичний timestamp,
- ✓ measurement ("sensor-data"),
- ✓ значення полів (temperature, humidity),
- ✓ опціональні теги (deviceId).

Завдання

1. Створіть новий ресурс IoT Hub, назва починалася з вашого прізвища.

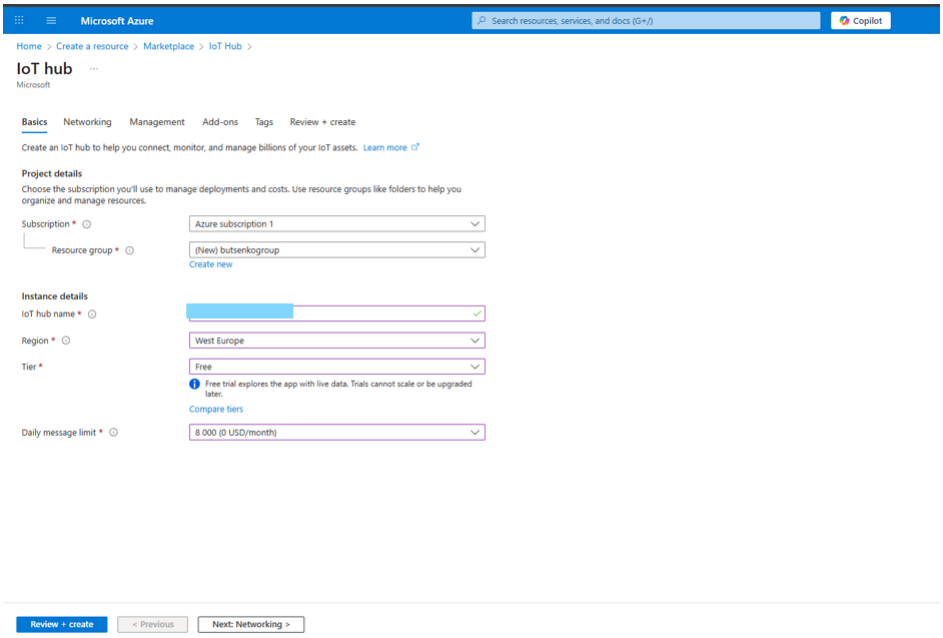


Рис. 1. Створення ресурсу IoT Hub

2. Відкрийте створений IoT Hub. У розділі Devices → + New device зареєструйте новий пристрій. Збережіть Primary Connection String вашого пристрою.

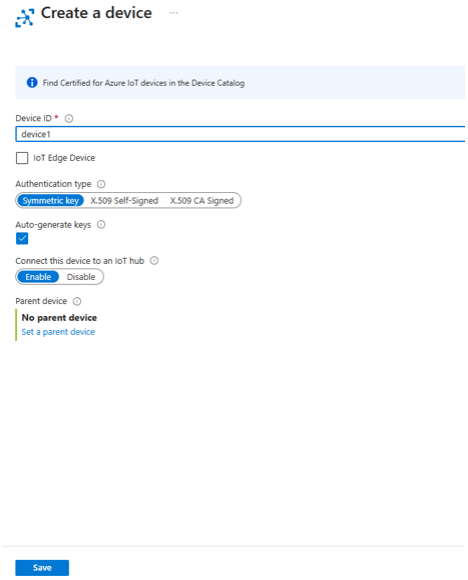


Рис. 2. Реєстрація приладу IoT

3. Відкрийте емулятор Raspberry Pi IoT: https://azure-samples.github.io/raspberry-pi-web-simulator/

4. Додайте Device Connection String у змінну connectionString у коді симулятора.

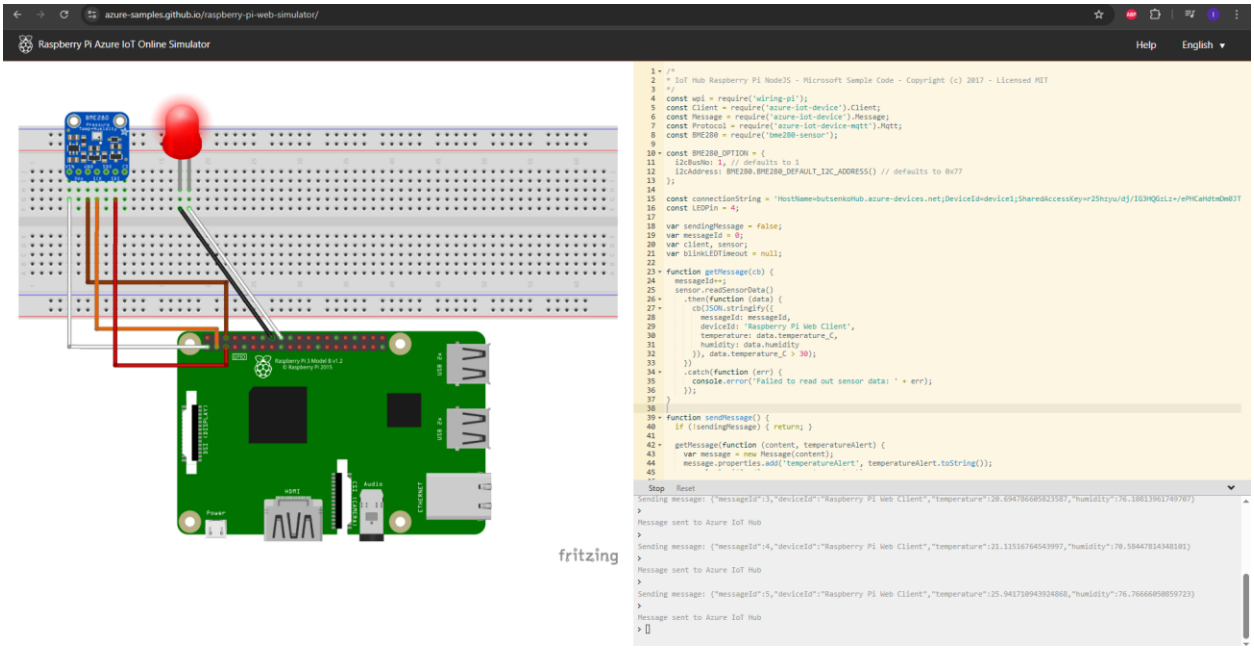


Рис. 3. Налаштування емулятор Raspberry Pi

5. Запустіть симуляцію, щоб пристрій почав надсилати повідомлення (температуру, вологість тощо) у IoT Hub.

6. Перейдіть за посиланням https://cloud2.influxdata.com/. Зареєструйтесь.

7. Створіть Bucket (сховище) з назвою, що починається з вашого прізвища.

8. Створіть API Token у розділі Load Data → API Tokens та збережіть його.

Використайте параметри підключення:

INFLUXDB_URL = "https://us-east-1-1.aws.cloud2.influxdata.com" INFLUXDB_TOKEN = <token для вашої сесії Load Data/API TOKEN> INFLUXDB_ORG = <назва організації з якої ви зареєстровані< INFLUXDB_BUCKET = назва сховища EVENT_HUB_CONNECTION_STRING = IoT Hub → Built-in endpoints → Event Hub-compatible endpoint EVENT_HUB_NAME = Event Hub-compatible name CONSUMER_GROUP = "$Default"

9. Напишіть код на Python, який буде:

- ✓ Отримувати повідомлення з IoT Hub (через Event Hub endpoint);
- ✓ Читати дані сенсорів (температура, вологість);
- ✓ Записувати їх у InfluxDB у таблицю (measurement) "sensor-data".

10. Перевірка результатів: Перегляньте дані у вебінтерфейсі InfluxDB (Data Explorer → ваш bucket → measurement sensor-data).

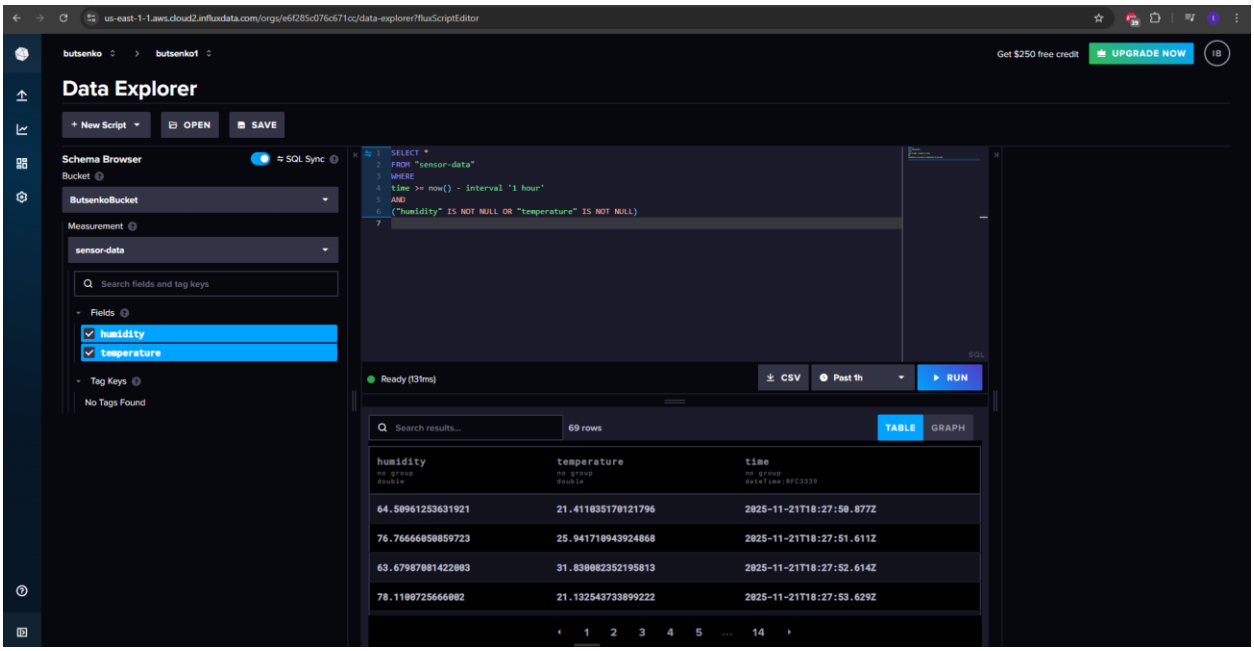


Рис. 4. Дані приладу IoT, збережені в таблиці InfluxDB

11. Підготуйте звіт, який включає: програму, знімки екрана (кроки 5, 10). Відповіді на контрольні запитання.

Контрольні запитання

1. Що таке Azure IoT Hub і яку роль він виконує в архітектурі IoT-системи?
2. Яке призначення Event Hub-compatible endpoint у IoT Hub?
3. Які типи даних зазвичай зберігаються в InfluxDB?
4. Чому InfluxDB підходить для роботи з часовими рядами?
5. Які основні компоненти має запит у InfluxDB (bucket, organization, token)?
6. Як можна візуалізувати отримані дані з InfluxDB?

Список літератури

1. Sreeram PK. Azure Serverless Computing Cookbook: Build and monitor Azure applications hosted on serverless architecture using Azure functions. Packt Publishing Ltd; 2020 Jun 19.

2. Seara DA, Milano F, Dominici D. Microsoft Azure Data Solutions-An Introduction. Microsoft Press; 2021 Jul 14.

3. L'Esteve RC. The Definitive Guide to Azure Data Engineering: Modern ELT, DevOps, and Analytics on the Azure Cloud Platform. Apress; 2021.

4. Seara DA, Milano F. Exam Ref DP-900 Microsoft Azure Data Fundamentals. Microsoft Press; 2021 Mar 12.

5. Stewart, Blaize. *Architecting IoT Solutions on Azure*. " O'Reilly Media, Inc.", 2024.

6. Соловей О. (2025). Сучасні технологічні рішення зберігання даних у проєктах міського будівництва. Управління розвитком складних систем, (63), 167–173. https://doi.org/10.32347/2412-9933.2025.63.167-173.

7. Olga Solovei, Tetiana Honcharenko, "Leveraging Sensitivity Analysis for Configurable Kafka Clusters: A Multi-Objective Model to Minimize Latency", International Journal of Intelligent Systems and Applications(IJISA), Vol.17, No.4, pp.25-39, 2025. DOI:10.5815/ijisa.2025.04.03

Інформаційні ресурси в Інтернет

1. "Azure Storage Actions documentation" за посиланням: https://learn.microsoft.com/en-us/azure/storage-actions/storage-tasks/

2. "Azure Logic Apps documentation" за посиланням: https://learn.microsoft.com/en-us/azure/logic-apps/

3. "Azure File Sync documentation" за посиланням: https://learn.microsoft.com/en-us/azure/storage/file-sync/?utm_source=chatgpt.com

4. "Azure Table storage documentation" за посиланням: https://learn.microsoft.com/en-us/azure/storage/tables/?utm_source=chatgpt.com

5. InfluxDB 3 Core documentation за посиланням: https://docs.influxdata.com/influxdb3/core/

Навчально-методичне видання

GRID-системи та хмарні технології

Методичні вказівки
до виконання лабораторних робіт
для здобувачів другого (магістерського) рівня вищої освіти
спеціальностей F2 (121) «Інженерія програмного забезпечення»,
F6 (126) «Інформаційні системи та технології»

Укладач **Соловей** Ольга Леонідівна

Комп’ютерне верстання *А.П. Селівестрової*

Ум. друк. арк. 2,09. Обл.-вид. арк. 2,25
Електронний документ. Вид № 157/V-25

Виконавець і виготовлювач

Київський національний університет будівництва і архітектури
Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб’єктів
видавничої справи ДК № 808 від 13.02.2002 р.