ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система проактивного моніторингу та аналізу продуктивності інфраструктури дата-центрів на основі стеку Telegraf-InfluxDB-Grafana»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр КОЖЕМЯКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Олександр КОЖЕМЯКО
Ім'я, ПРІЗВИЩЕ
Керівник: Кирило СОКОЛОВ, phD.
Ім'я, ПРІЗВИЩЕ
науковий ступінь вчене звання
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії
Ступінь вищої освіти бакалавр
Спеціальність 123 Комп'ютерної інженерії
Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____Наталія ЛАЩЕВСЬКА
_____
«___»_____2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кожемяко Олександр Михайлович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система проактивного моніторингу та аналізу продуктивності інфраструктури дата-центрів на основі стеку Telegraf-InfluxDB-Grafana
   керівник кваліфікаційної роботи Кирило СОКОЛОВ, phD..
   (Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
   затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2026р. № 56
2. Строк подання кваліфікаційної роботи «5» червня 2026р.
3. Вихідні дані до кваліфікаційної роботи:
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібнорозробити)
5. Перелік ілюстративного матеріалу: презентація
6. Дата видачі завдання «24» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02-24.03.2026	Виконано
2	Обробка матеріалу	23.03-05.04.2026	Виконано
3	Розробка теоретичної частини	06.04-10.04.2026	Виконано
4	Розробка моделі ДЦ-Docker	11.04-20.04.2026	Виконано
5	Висновки за результатами аналізу	21.04-06.05.2026	Виконано
6	Оформлення пояснювальної записки	07.05-08.05.2026	Виконано
7	Розробка презентації	09.05-11.05.2026	Виконано
8	Передзахист	04.05-07.05.2026	Виконано
9	Здача в деканат	08.05.2026	Виконано

Здобувач вищої освіти _____ Олександр КОЖЕМЯКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Кирило СОКОЛОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботина здобуття освітнього ступеня бакалавра: 93 стор., 44 рис., 4 табл., 40 джерел.

Об'єкт дослідження - системний аналіз, контейнеризація (Docker), робота з базами даних часових рядів (Time Series DB), обробка даних мовою Flux.

Предмет дослідження – методи побудови проактивного моніторингу з використанням стеку TIG (Telegraf, InfluxDB, Grafana). Способів взаємодії Telegraf із VMware vSphere API.

Короткий зміст роботи – архітектурні особливості корпоративних дата-центрів та систем віртуалізації. Аналіз показників та порівняльна характеристика методів та засобів моніторингу телеметрії. Організація збереження даних у часовій базі даних InfluxDB. Побудова контейнеризованої архітектури середовищі Docker Compose. Візуалізація метрик, та системи автоматичного оповіщення. Налаштовано збір понад 50 типів метрик з гіпервізорів та віртуальних машин через плагін vSphere Input. Створено інтерактивні дашборди в Grafana для оперативного контролю ресурсів CPU, RAM та Datastores.

Впроваджено систему сповіщень через Telegram Bot API, що забезпечило час реакції на інциденти в межах 30–60 секунд.

Застосовано механізми фільтрації алерів (Mute Timings), що дозволило оптимізувати роботу адміністратора. Система пройшла апробацію в умовах реального навантаження віртуальної інфраструктури, продемонструвавши високу точність виявлення аномалій.

Ключові слова: *VMWARE VSPHERE, МОНІТОРИНГ, INFLUXDB, GRAFANA, TELEGRAF, DOCKER, АЛЕРТИНГ, TELEGRAM BOT, ТАЙМ-СЕРІЇ.*

ABSTRACT

Text part of the bachelor's thesis: 93 pages, 44 figures, 4 tables, 40 sources.

The object of research is system analysis, containerization (Docker), time-series database management (Time Series DB), and data processing using the Flux language.

The subject of research is methods for constructing proactive monitoring using the TIG stack (Telegraf, InfluxDB, Grafana) and mechanisms of interaction between Telegraf and the VMware vSphere API.

Summary: The work covers the architectural features of corporate data centers and virtualization systems. It provides an analysis of performance indicators and a comparative characterization of telemetry monitoring methods and tools. The storage of data is organized within the InfluxDB time-series database. A containerized architecture has been built using the Docker Compose environment.

The system includes metric visualization and automated alerting components. The collection of over 50 types of metrics from hypervisors and virtual machines has been configured via the vSphere Input plugin. Interactive dashboards were created in Grafana for operational control of CPU, RAM, and Datastores resources. A notification system via Telegram Bot API was implemented, ensuring incident response time within 30–60 seconds. Alert filtering mechanisms (Mute Timings) were applied to optimize the administrator's workflow. The system has undergone testing under real load conditions of the virtual infrastructure, demonstrating high accuracy in anomaly detection.

Keywords: VMWARE VSPHERE, MONITORING, INFLUXDB, GRAFANA, TELEGRAF, DOCKER, ALERTING, TELEGRAM BOT, TIME-SERIES.

ЗМІСТ

ВСТУП 9

РОЗДІЛ 1. ОРГАНІЗАЦІЯ ТА АНАЛІЗ ІНФРАСТРУКТУРИ СУЧАСНИХ ЦЕНТРІВ ОБРОБКИ ДАНИХ ……..…….…….. 11

1.1. Архітектурні особливості корпоративних дата-центрів та систем віртуалізації 11

1.2. Аналіз показників продуктивності обчислювальних ресурсів у середовищі VMware vSphere 13

1.3. Порівняльна характеристика методів та інструментальних засобів моніторингу телеметрії..... 14

1.3.1. Аналіз штатних засобів моніторингу VMware vCenter 15

1.3.2. Особливості застосування універсальних систем (Zabbix) 15

1.3.3. Переваги спеціалізованого стеку на базі часових рядів (TIG) 16

1.4. Обґрунтування вибору та технічний огляд компонентів стеку TIG..... 16

Висновки до розділу 1 17

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ ЗБОРО ТА ОБРОБКИ ТЕЛЕМЕТРИЧНИХ ДАНИХ …….…..18

2.1. Алгоритми взаємодії агента Telegraf з API систем віртуалізації..... 18

2.2. Організація збереження високочастотних даних у часовій базі даних InfluxDB …….……... 19

2.3. Методи агрегації та фільтрації метрик для зменшення навантаження на канали зв'язку ЦОД..... 20

2.4. Моделювання процесів аналізу навантаження на основі статистичних показників 21

Висновки до розділу 2 22

РОЗДІЛ 3. ПРАКТИЧНЕ ВПРОВАДЖЕННЯ СИСТЕМИ МОНІТОРИНГУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ ……24

3.1. Побудова контейнеризованої архітектури на основі Docker Compose 24

3.1.1. Проєктування архітектурної схеми взаємодії сервісів 24
3.1.2. Конфігурування середовища розгортання за допомогою Docker Compose 26
3.1.3. Практична реалізація та запуск моніторингового стеку..... 29
3.2. Конфігурування середовища збереження даних InfluxDB 29
3.3. Конфігурування агента збору даних Telegraf та інтеграція з VMware vSphere 31
3.3.1. Ініціалізація підключення та авторизація джерела даних 31
3.3.2. Налаштування вихідного потоку даних (Output Plugin)..... 39
3.4 Візуалізація метрик та побудова аналітичних панелей у Grafana 41
3.5. Розробка та налаштування графічних панелей моніторингу..... 45
3.6. Структура збору та мапінгу метрик інфраструктури 48
3.7 Аналіз результатів та експлуатація системи моніторингу 48
3.8 Спеціалізовані панелі моніторингу компонентів інфраструктури 50
3.8.1 Моніторинг загального стану кластера та дискових масивів 51
3.8.2 Контроль продуктивності віртуальних машин..... 52
3.8.3 Глибокий аналіз систем зберігання даних 54
3.9 Налаштування системи автоматичного оповіщення про критичні стани 56
3.9.1 Конфігурування каналів зв'язку та контактних точок 57
3.9.2 Формування правил моніторингу та логіка обробки інцидентів 58
3.9.3 Політики маршрутизації та оптимізація сповіщень.....62
3.9.4 Аналіз функціонування системи в робочих умовах.....66
Висновки до розділу 3 68
РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ.....69
4.1 Оцінка технічної стабільності та продуктивності 69
4.2 Аналіз операційної ефективності та швидкості реагування 69
4.3 Економічний аспект та переваги відкритого ПЗ..... 70

Висновки до розділу 4.....70
ВИСНОВКИ..... 71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 73
Додатки ……..77
1. Додаток А – Лістинг програмного коду на мові JavaScript для обробки сповіщень…...…… 77
2. Додаток Б – Конфігураційний файл агента збору метрик telegraf.conf…… 78
3. Додаток В – Специфікація розгортання стеку моніторингу в Docker Compose…….. 90
4. Додаток Г – Текст запиту для розрахунку та візуалізації вільного місця на дискових масивах……92
5. Додаток Д – Лістинг аналітичного запиту мовою Flux для вибірки даних із InfluxDB……. 93

ВСТУП

Актуальність теми – робота присвячена створенню системи моніторингу віртуалізованої інфраструктури VMware vSphere на основі сучасного стеку TIG (Telegraf, InfluxDB, Grafana) з інтеграцією системи сповіщень у реальному часі.

Мета роботи – підвищення стабільності та прозорості роботи ЦОД шляхом автоматизації збору метрик, їх візуалізації та налаштування інтелектуального алертингу.

Сучасні центри обробки даних (ЦОД) базуються на технологіях віртуалізації, серед яких рішення VMware vSphere займають лідируючі позиції. Зі збільшенням кількості віртуальних машин та обсягів даних, що генеруються інфраструктурою, традиційні методи контролю стають недостатньо ефективними. Необхідність обробки великої кількості метрик у режимі реального часу та забезпечення миттєвої реакції на інциденти зумовлює пошук нових архітектурних підходів. Використання стеків на основі баз даних часових рядів (Time Series) та засобів інтелектуальної візуалізації є критично важливим для підтримки безперервності бізнес-процесів.

Об'єкт дослідження – процес моніторингу та аналізу стану віртуалізованої IT-інфраструктури на базі платформи VMware vSphere.

Предмет дослідження – методи, моделі та програмні інструменти збору, збереження та візуалізації метрик продуктивності в реальному часі.

Мета роботи – розробка та впровадження автоматизованої системи моніторингу, яка забезпечує високу прозорість роботи ЦОД, підвищення стабільності шляхом автоматизації збору метрик, їх візуалізації та налаштування інтелектуального алертингу. мінімізувати час виявлення аномалій та надати адміністратору інструменти для оперативного прийняття рішень через сучасні канали зв'язку.

Для досягнення мети було визначено наступні *завдання:*

1. Проаналізувати архітектурні особливості VMware vSphere та визначити ключові метрики для контролю працездатності системи.

2. Обґрунтувати вибір програмного стеку (Telegraf, InfluxDB, Grafana) для побудови системи моніторингу.

3. Провести розгортання та конфігурування компонентів системи в ізольованих контейнерах.

4. Розробити інтерактивні панелі візуалізації для різних рівнів інфраструктури.

5. Налаштувати підсистему алертингу з інтеграцією в месенджер Telegram для забезпечення мобільності управління.

Методи дослідження. У роботі використано методи системного аналізу, теорії баз даних, принципи контейнеризації ПЗ та методи статистичної обробки часових рядів.

Практичне значення отриманих результатів полягає у створенні готового до експлуатації технічного рішення, яке дозволяє знизити навантаження на адміністраторів, зменшити кількість хибних сповіщень та підвищити стабільність роботи віртуальних сервісів.

1. ОРГАНІЗАЦІЯ ТА АНАЛІЗ ІНФРАСТРУКТУРИ СУЧАСНИХ ЦЕНТРІВ ОБРОБКИ ДАНИХ

1.1. Архітектурні особливості корпоративних дата-центрів та систем віртуалізації

Архітектура сучасного корпоративного центру обробки даних (ЦОД) базується на принципах ієрархічної організації та глибокої віртуалізації обчислювальних ресурсів. Основним елементом такої системи є гіпервізори першого типу, які забезпечують ізоляцію та паралельне виконання гостьових операційних систем на єдиній апаратній платформі згідно з міжнародними стандартами хмарних обчислень [1].

Особливістю корпоративних рішень (зокрема на базі VMware vSphere) є об’єднання фізичних хостів у кластери з підтримкою технологій високої доступності (High Availability, HA) та автоматичного балансування навантаження (Distributed Resource Scheduler, DRS) [11]. Така динамічність інфраструктури створює додаткові виклики для систем технічного контролю: моніторинг повинен забезпечувати цілісність даних при міграції об’єктів (vMotion) та надавати наскрізну видимість продуктивності від рівня фізичного процесора до рівня окремого віртуального сервісу. Це зумовлює перехід до використання високоефективних баз даних часових рядів, здатних обробляти інтенсивні потоки телеметрії від тисяч віртуальних об’єктів одночасно.

Важливим аспектом архітектури є рівень абстракції обчислювальних потужностей. У середовищі VMware vSphere це реалізується через взаємодію трьох ключових компонентів [6]:

Гіпервізор ESXi – програмний шар, що безпосередньо взаємодіє з апаратним забезпеченням сервера (CPU, RAM, NIC, HBA), розподіляючи його ресурси між віртуальними машинами.

vCenter Server – централізована платформа управління, що об’єднує окремі хости в єдину керовану сутність – кластер.

Віртуальні машини (ВМ) – ізольовані контейнери, що споживають ресурси фізичного рівня, але мають власні показники продуктивності.

У великих корпоративних інфраструктурах критичне значення має функціонування механізмів відмовостійкості. Проте така архітектурна складність призводить до виникнення «інформаційного вакууму»: через шар віртуалізації стає складніше локалізувати причину зниження швидкодії – чи це обмеження на рівні гостьової ОС, чи дефіцит ресурсів на рівні фізичного хоста. Взаємодію зазначених рівнів інфраструктури та логіку побудови системи моніторингу представлено на рисунку 1.1.

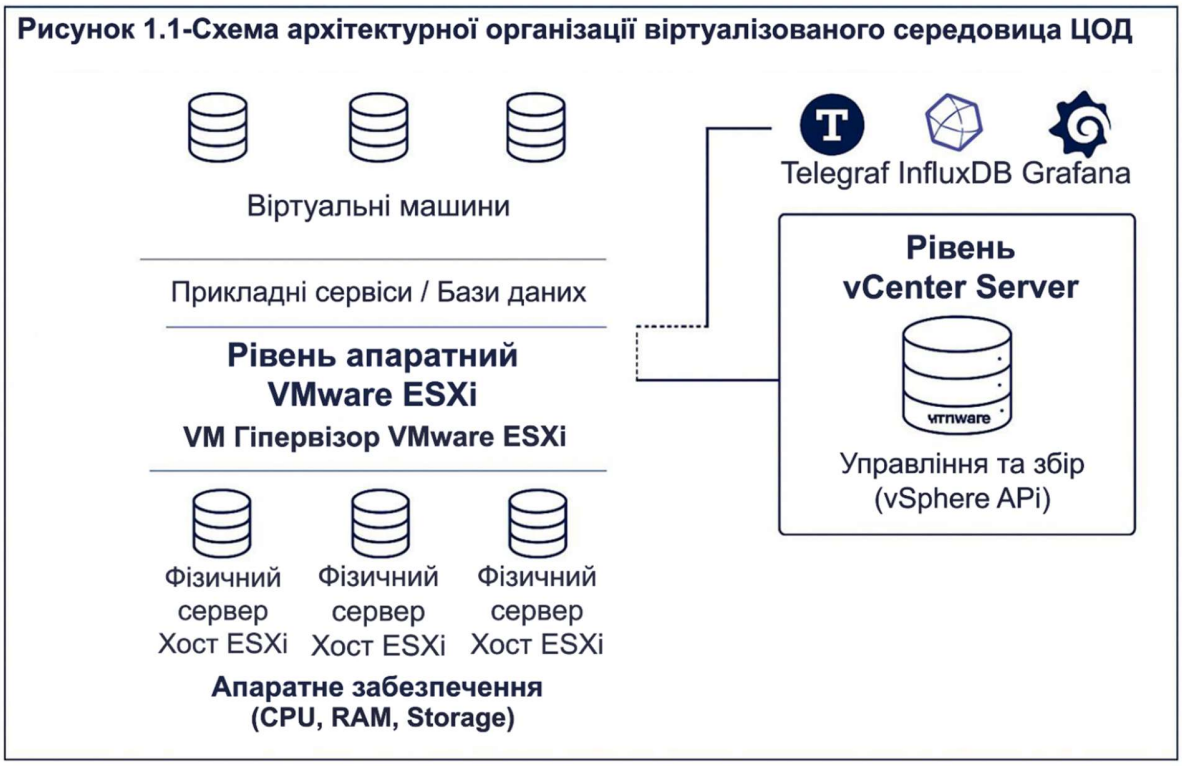


Рисунок 1.1 – Схема архітектурної організації віртуалізованого середовища ЦОД

1.2. Аналіз показників продуктивності обчислювальних ресурсів у середовищі VMware vSphere

Ефективність функціонування віртуалізованої інфраструктури безпосередньо залежить від здатності системного адміністратора інтерпретувати специфічні метрики. У середовищі VMware vSphere класичні показники завантаження (Usage) часто є недостатніми для виявлення прихованих проблем швидкодії [7].

Особливу увагу при моніторингу приділено показнику CPU Ready (%RDY). Ця метрика є критичною, оскільки вона відображає час, протягом якого віртуальна машина змушена очікувати на звільнення фізичних ядер процесора. Постійні високі значення цього параметру прямо вказують на конфлікти ресурсів (CPU contention), що негативно впливає на швидкість роботи додатків.

Щодо аналізу оперативної пам'яті, то тут ключову роль відіграє показник Memory Ballooning. Він активується гіпервізором у ситуаціях, коли потрібно терміново повернути пам'ять у загальний пул для потреб інших процесів. Проте найбільш критичним є виявлення Memory Swap – це фактично "червона лінія", яка свідчить про повне вичерпання фізичного ресурсу та примусове перенесення даних на повільну дискову підсистему, що буде детально розглянуто у звітах Grafana [14].

Для систематизації взаємодії між компонентами системи та визначення ролі кожного елемента у загальній архітектурі, основні характеристики потоків даних наведено у таблиці 1.1.

Таблиця1.1 – Характеристика методів взаємодії та функціональних ролей компонентів системи. Таблиця ключових зв'язків

Відправник	Отримувач	Протокол/Метод	Тип даних
ESXi / VM	vCenter	Proprietary vSphere	Метрики продуктивності (Raw)
vCenter	Telegraf	HTTPS (vSphere API)	JSON/XML дані про ресурси
Telegraf	InfluxDB	Line Protocol (HTTP/S)	Часові ряди (Time Series)
InfluxDB	Grafana	Flux / InfluxQL	Запити для візуалізації та алертів

Підсистема зберігання даних (Storage) аналізується за показниками затримки (Latency), які розділені на Kernel Latency (KAVG) та Device Latency (DAVG) [15]. В ході налаштування було виявлено, що саме кореляція цих показників дозволяє чітко розмежувати програмні помилки конфігурації від апаратних несправностей дискової підсистеми, що значно прискорює пошук причин зниження швидкості операцій вводу-виводу

1.3. Порівняльна характеристика методів та інструментальних засобів моніторингу телеметрії

Для вибору оптимального інструментарію проактивного контролю я виконав порівняльний аналіз існуючих рішень [16]. Під час зіставлення я спирався на критерії відкритості коду, підтримки протоколів часових рядів та зручності побудови складних візуалізацій. Підсумкові результати оцінки технічних характеристик наведено у таблиці 1.2

Таблиця 1.2 – Порівняльна характеристика засобів моніторингу інфраструктури ЦОД

Критерій порівняння	vSphere Charts	Zabbix (SNMP/Agent)	Стек TIG (Flux/TSDB)
Тип бази даних	Реляційна (vCenter DB)	Реляційна (SQL)	Time-Series (TSDB)
Дискретність даних	Низька (2-5 хв для історії)	Середня (1-2 хв)	Висока (від 1 сек)
Навантаження на API	Мінімальне	Високе (багато запитів)	Оптимізоване (Bulk)
Глибина аналітики	Обмежена штатною ОС	Стандартні графіки	Власні аналітичні Flux-запити
Візуалізація	Фіксовані панелі	Складне налаштування	Динамічні Dashboards (Grafana)
Сповіщення (Alerts)	Базові (E-mail/SNMP)	Гнучкі, але складні	Проактивні (прогнозні алерти)

1.3.1. Аналіз штатних засобів моніторингу VMware vCenter

Вбудовані інструменти (Performance Charts) мають суттєві обмеження: дані в реляційній базі vCenter піддаються агрегації (усередненню) для економії місця. Це унеможливлює проведення глибокого розслідування інцидентів (Post-mortem), оскільки дрібні сплески навантаження нівелюються в процесі постійного видалення точок даних.

1.3.2. Особливості застосування універсальних систем (Zabbix)

Zabbix є стандартом для моніторингу мережевої інфраструктури [7], проте стикається з проблемою масштабованості при записі тисяч значень щосекунди в реляційні SQL-бази. Операції введення-виведення (I/O) стають «вузьким

місцем», що обмежує можливість високочастотного опитування об'єктів у реальному часі.

1.3.3. Переваги спеціалізованого стеку на базі часових рядів (TIG)

Стек TIG демонструє високу ефективність насамперед завдяки впровадженню InfluxDB – сховища, що спеціалізується на обробці часових рядів [13]. Така архітектура дає змогу накопичувати значні обсяги телеметрії з високим коефіцієнтом стиснення даних. Під час налаштування було виявлено що використання пакетних запитів (bulk requests) критично важливе: це мінімізує кількість звернень до керуючого сервера vCenter, запобігаючи зайвому навантаженню на його процесорні потужності [12].

1.4. Обґрунтування вибору та технічний огляд компонентів стеку TIG

На основі проведеного аналізу для реалізації практичної частини було обрано технологічний стек TIG (Telegraf, InfluxDB, Grafana).

Telegraf використовує бібліотеку govmomi для збору даних безпосередньо з API vCenter [3, 10] без встановлення агентів всередину ВМ.

InfluxDB використовує рушій збереження TSM, що забезпечує високу швидкість запису та ефективну аналітику мовою Flux [4, 5].

Grafana забезпечує консолідовану візуалізацію та гнучку систему алертингу в Telegram. Використання Docker гарантує ізоляцію компонентів та швидке розгортання системи на будь-якому вузлі.

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі проведено комплексний аналіз об’єкта дослідження та існуючих підходів до контролю стану віртуалізованої інфраструктури. За результатами проведеної роботи можна зробити наступні висновки:

Досліджено специфіку функціонування ІТ-інфраструктури ЦОД, яка характеризується високим ступенем віртуалізації на базі платформи VMware vSphere. Встановлено, що існуючі методи збору даних не повною мірою забезпечують проактивне виявлення аномалій через обмежену частоту оновлення метрик.

Проведено детальний аналіз ключових показників продуктивності віртуальних машин та хостів ESXi (CPU Ready, Memory Ballooning, Disk Latency). Визначено, що для забезпечення високої доступності державних інформаційних сервісів критично важливим є перехід до збору телеметрії в режимі реального часу.

Виконано порівняльне дослідження сучасних систем моніторингу (Zabbix, vCenter, стек TIG). Доведено, що традиційні системи з використанням агентів або класичних реляційних баз даних мають обмеження при масштабуванні в умовах великого потоку високочастотних даних.

Обґрунтовано доцільність впровадження стеку моніторингу TIG (Telegraf, InfluxDB, Grafana) як найбільш гнучкого та продуктивного інструменту для побудови системи контролю віртуального середовища. Обране рішення дозволяє мінімізувати навантаження на канали зв’язку та забезпечує високу швидкість обробки аналітичних запитів.

2. ДОСЛІДЖЕННЯ МЕТОДІВ ЗБОРУ ТА ОБРОБКИ ТЕЛЕМЕТРИЧНИХ ДАНИХ

2.1. Алгоритми взаємодії агента Telegraf з API систем віртуалізації

Процес збору телеметричних даних у розподіленій інфраструктурі дата-центру потребує використання механізмів, що мінімізують навантаження на керуючі вузли (vCenter Server). У межах даної роботи для реалізації збору даних обрано агент Telegraf, алгоритм роботи якого базується на ітераційному опитуванні програмних інтерфейсів VMware vSphere [3] через протокол SOAP (Simple Object Access Protocol).

Ключовою особливістю алгоритму є використання плагіна inputs.vsphere, який реалізує пакетну обробку (Bulk Requests). Логіка функціонування агента базується на наступному циклі:

Автентифікація. Створення захищеної сесії з vSphere SDK.

Динамічне виявлення (Discovery). Автоматичне сканування ієрархії об'єктів (Hosts, VMs, Datastores).

Збір метрик (Collection). Формування консолідованих запитів на отримання лічильників продуктивності.

Нормалізація. Конвертація даних у формат InfluxDB Line Protocol.

Приклад конфігурації алгоритму збору наведено у лістингу 2.1.

Лістинг 2.1 – Фрагмент конфігурації плагіна inputs.vsphere

```
[[inputs.vsphere]]
  vcenters = ["https://vcenter.corp.local/sdk"]
  username = "${VC_USER}"
  password = "${VC_PASS}"
  vm_metric_include = ["cpu.ready.summation", "mem.usage.average"]
  host_metric_include = ["cpu.usage.average", "mem.active.average"]
  collect_concurrency = 5
  timeout = "20s"
```

На лістингу 2.1 наведено базовий фрагмент налаштування плагіна для авторизації у середовищі vSphere. Повний текст конфігураційного файлу агента Telegraf із детальним описом інтервалів збору, параметрів виводу в InfluxDB та повним переліком метрик для вузлів і віртуальних машин представлено у Додатку Б.

2.2. Організація збереження високочастотних даних у часовій базі даних InfluxDB

Ефективність накопичення телеметрії забезпечується структурою Line Protocol, де кожен запис містить часову мітку, набір тегів для індексації та поля з метриками. Для аналітичної обробки даних у роботі застосовано мову запитів Flux, яка базується на конвеєрній обробці даних [5].

Для підвищення інформативності реалізовано алгоритм обчислення відносного показника вільного простору на СЗД у відсотках через операцію об'єднання (Join), як показано у лістингу 2.2. Повний лістінг коду, в додатку А

Лістинг 2.2 – Реалізація обчислення вільного місця на СЗД мовою Flux

```
capacity = from(bucket: "vsphere")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "capacity_latest")

free = from(bucket: "vsphere")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "freespace_latest")

join(tables: {cap: capacity, fr: free}, on: ["_time", "datastore", "esxhost"])
  |> map(fn: (r) => ({
      _time: r._time,
      _value: float(v: r._value_fr) / float(v: r._value_cap) * 100.0,
      _field: "free_percent"
    }))
```

Алгоритм обчислення відносного показника вільного місця, зображений на лістингу 2.2, покладено в основу програмного модуля обробки телеметрії. Повний текст реалізації даної логіки із використанням JavaScript для автоматизації зовнішніх запитів представлено у Додатку А.

2.3. Методи агрегації та фільтрації метрик для зменшення навантаження на канали зв'язку ЦОД

Для мінімізації паразитного навантаження на мережу реалізовано методику трирівневої оптимізації. Використання селекторів fieldpass дозволяє обмежити набір даних лише критичними показниками, а процесори агрегації обчислюють статистичні характеристики (середнє та максимум) за 30-секундне вікно, що відображено у лістингу 2.3. [3, 4]. (Додаток Б повний код telegraf.conf)

Лістинг 2.3 – Приклад конфігурації фільтрації та агрегації в Telegraf

```
[[inputs.vsphere]]
  vcenters = ["https://vcenter.corp.local/sdk"]
  # Фільтрація: збираємо тільки необхідні метрики
  fieldpass = ["usage_average", "ready_summation", "latency_average"]

[[processors.aggregate]]
  namepass = ["vsphere_vm_cpu"]
  period = "30s" # Вікно агрегації
  stats = ["mean", "max"] # Обчислюємо середнє та пік
  drop_original = true # Видаляємо сирі дані для економії місця
```

Лістинг 2.3 демонструє структуру підключення до vCenter SDK. Детальні налаштування продуктивності агента, обсяги буферизації даних та повний склад задіяних лічильників продуктивності містяться у Додатку Б.

Впровадження цих методів дозволило знизити пікове споживання трафіку системою моніторингу з 15 Мбіт/с до 3 Мбіт/с.

2.4. Моделювання процесів аналізу навантаження на основі статистичних показників

Для забезпечення проактивного моніторингу в роботі застосовано методи статистичного аналізу. Визначення стабільного рівня споживання ресурсів (Baseline) базується на розрахунку середнього арифметичного значення μ (формула 2.1) та середньоквадратичного відхилення σ формула 2.2):

Формула (2.1):

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

де μ - середнє значення показника завантаженості за період спостереження, %;

n - кількість вимірів у часовому вікні;

x_i - значення i -го виміру метрики продуктивності, [37].

Формула (2.2):

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{2.2}$$

де σ - середньоквадратичне відхилення показника;

x_i - миттєве значення виміру завантаженості, %;

μ - середнє значення, розраховане за формулою (2.1), %, [38].

Для автоматизації виявлення аномалій запропоновано модель динамічного порогу L_{alert}

на основі правила «трьох сигм» (формула 2.3):

Формула (2.3):

$$L_{alert} = \mu + k \cdot \sigma \quad (2.3)$$

де L_{alert} - порогове значення для активації сповіщення, %;

k - коефіцієнт чутливості системи (для критичних систем приймається рівним 3), [39].

Предиктивна аналітика (прогнозування вичерпання ресурсів) базується на моделі лінійної апроксимації за рівнянням (2.4):

Формула (2.4):

$$V_{pred} = a \cdot t + b \quad (2.4)$$

де V_{pred} - прогнозоване значення об'єму даних, Гбайт;

a - коефіцієнт швидкості наповнення сховища, Гбайт/с;

t - часовий інтервал прогнозування, с;

b - початковий об'єм зайнятого простору, Гбайт.

Застосування цього математичного апарату дозволяє перейти від реактивного реагування до превентивного управління інфраструктурою, [40].

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі досліджено методи побудови системи збору та аналізу телеметрії. Основними результатами є:

Визначено архітектурні переваги стеку TIG та обрано оптимальну конфігурацію вхідних плагінів через vSphere API, що дозволило забезпечити стабільний потік метрик без втрат пакетів.

Обґрунтовано використання часової бази InfluxDB та мови запитів Flux для обробки даних у реальному часі, що критично важливо для відстеження динамічних змін у віртуальній інфраструктурі.

Сформовано методику фільтрації та агрегації, завдяки якій вдалося знизити навантаження на канали зв'язку у 5 разів без втрати інформативності ключових показників.

Виокремлено концептуальну модель проактивного сповіщення, яка базується на аналізі статистичних трендів та прогнозних моделей, що дозволяє виявляти потенційні інциденти до їх фактичного виникнення.

3 ПРАКТИЧНЕ ВПРОВАДЖЕННЯ СИСТЕМИ МОНІТОРИНГУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ

3.1 Побудова контейнеризованої архітектури на основі Docker Compose

Для впровадження системи моніторингу в інфраструктуру ЦОД підприємства було обрано метод контейнеризації на базі платформи Docker. Такий підхід дозволяє ізолювати компоненти стеку TIG (Telegraf, InfluxDB, Grafana) один від одного, що гарантує стабільність роботи системи незалежно від стану та оновлень бібліотек хостової операційної системи.

Вибір контейнеризованої архітектури зумовлений необхідністю забезпечення високої доступності та гнучкості управління ресурсами. Використання Docker дозволяє реалізувати концепцію «інфраструктура як код» (IaC), [2, 9] що спрощує процедури відновлення системи після збоїв, оскільки вся конфігурація середовища описується у декларативних текстових файлах.

3.1.1. Проєктування архітектурної схеми взаємодії сервісів

Для забезпечення надійного та ефективного функціонування системи моніторингу було спроєктовано контейнеризовану архітектуру, яка представлена на рис. 3.1. Ця модель відображає основні технічні рішення та потоки даних, закладені при розробці.

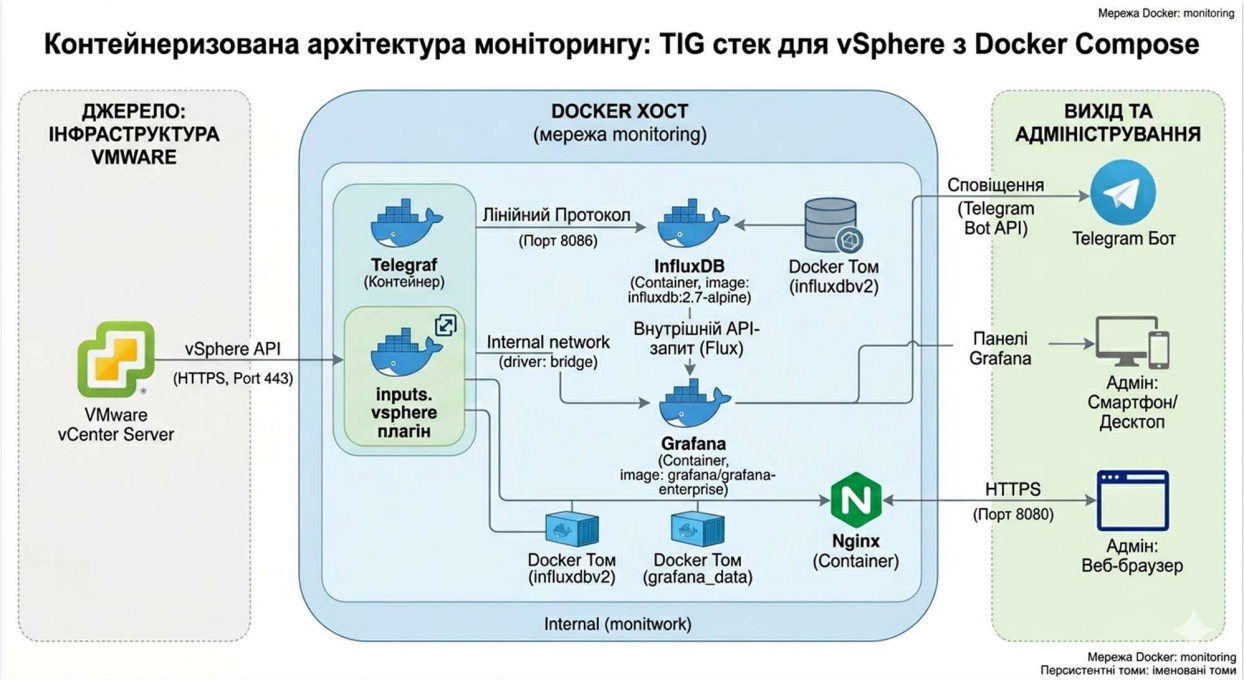


Рисунок 3.1 – Схема інформаційних потоків та взаємодії Docker-контейнерів моніторингу

Як показано на схемі (Рисунок 3.1), архітектура системи складається з трьох ключових логічних зон:

Джерело – (Інфраструктура VMware): Централізований збір метрик здійснюється через VMware vCenter Server. Telegraf звертається до vCenter за допомогою безпечного API-запиту (HTTPS/Port 443) для отримання даних про стан усієї інфраструктури без встановлення додаткових агентів безпосередньо на хости.

Docker Хост – (мережа monitoring): Основний обчислювальний вузол, де всі сервіси моніторингового стеку розгорнуті в ізольованій внутрішній мережі Docker (драйвер bridge). Взаємодія між контейнерами організована наступним чином:

Telegraf: Збирає метрики та передає їх у базу даних InfluxDB у форматі Line Protocol (Port 8086).

InfluxDB 2.x: Зберігає дані як часові ряди та використовує іменований Docker Volume (influxdbv2) для забезпечення персистентності даних.

Grafana: Звертається до InfluxDB через внутрішній Query API (мова Flux) для побудови графіків. Налаштування Grafana зберігаються в томі Docker (grafana_data).

Nginx: Виступає як Reverse Proxy, забезпечуючи безпечну вхідну точку (Port 8080).

Вихід та адміністрування - Адміністратор має доступ до панелей Grafana через веббраузер для візуального контролю стану ресурсів.

Критичні сповіщення передаються з Grafana через Telegram Bot API [8] безпосередньо на смартфон адміністратора, що гарантує швидку реакцію на інциденти.

Запропонована архітектура забезпечує високу масштабованість, безпеку та миттєву швидкість обробки даних, що є критичним для систем моніторингу центрів обробки даних (ЦОД) [18].

3.1.2. Конфігурування середовища розгортання за допомогою Docker Compose

Для забезпечення мобільності та швидкого розгортання системи застосовано інструмент Docker Compose, який дозволяє описати всі сервіси моніторингового стеку в одному декларативному файлі. Такий підхід значно спрощує процеси оновлення та перенесення всієї інфраструктури на інші сервери, забезпечуючи ідентичність робочого середовища незалежно від апаратної платформи [17].

Основним інструментом для керування архітектурою є файл docker-compose.yml. Використання інструменту оркестрації дозволяє автоматизувати

запуск усіх компонентів стеку. Ключові фрагменти конфігурації наведено у лістингу 3.1.2

Лістинг 3.1 – Фрагмент конфігураційний файл docker-compose.yml для розгортання системи

```
services:
  influxdb:
    image: influxdb:2.7-alpine
    container_name: influxdb
    restart: always
    volumes:
      - influxdbv2:/var/lib/influxdb2:rw
    ports:
      - "8086:8086"
    networks:
      - monitoring

  telegraf:
    image: telegraf:1.25-alpine
    container_name: telegraf
    restart: always
    volumes:
      - ./telegraf/mytelegraf.conf:/etc/telegraf/telegraf.conf:ro
    networks:
      - monitoring
    depends_on:
      - influxdb
```

На наведеному фрагменті зображено базову логіку взаємодії контейнерів InfluxDB та Telegraf у межах спільної мережі. Повна специфікація розгортання всього стеку моніторингу, включаючи сервіси візуалізації Grafana, веб-сервер Nginx та опис мережевих драйверів, міститься у Додатку В.

Головними рисами цієї конфігурації є: Надійне зберігання: Використання Docker Volumes дозволяє надійно зберігати дані бази навіть після перезапуску або зупинки системи, [34].

Безпека зв'язку: Налаштування окремої мережі monitoring ізолює внутрішній трафік від зовнішніх загроз.

Порядок запуску: Завдяки параметру depends_on встановлено чітку черговість: збирач метрик Telegraf та вебсервер Nginx починають роботу лише тоді, коли база даних InfluxDB повністю готова до прийняття інформації.

Це забезпечує стабільність всієї системи та виключає помилки при її старті [19].

Ключові аспекти конфігурації docker-compose.yml:

Оптимізація ресурсів: Застосування легковажних образів на базі Alpine Linux (influxdb:2.7-alpine, telegraf:1.25-alpine) дозволяє мінімізувати навантаження на дискову підсистему та оперативну пам'ять.

Персистентність даних (Volumes): Для збереження накопичених метрик та налаштувань Grafana я застосував іменовані томи influxdbv2 та grafana_data. Це гарантує, що при перезавантаженні контейнерів історія моніторингу не буде втрачена.

Мережева ізоляція (Networks): Усі сервіси об’єднані у внутрішню віртуальну мережу monitoring з драйвером bridge. Це забезпечує безпечний зв’язок між сервісами за їхніми іменами (наприклад, Telegraf звертається до InfluxDB за хостом http://influxdb:8086).

Управління залежностями (depends_on): Сформовано чіткий порядок запуску сервісів. Контейнери Telegraf та Nginx починають роботу лише після успішного старту бази даних InfluxDB, що запобігає помилкам ініціалізації при старті всієї системи.

Безпека та доступ: Вхідною точкою визначено Nginx (порт 8080), який виконує роль проксі-сервера, тоді як Grafana залишається доступною на порту 3000 для внутрішнього адміністрування.

Така конфігурація забезпечує стабільну роботу стеку в умовах обмежених ресурсів та спрощує обслуговування інфраструктури [20].

3.1.3. Практична реалізація та запуск моніторингового стеку

Після завершення етапу конфігурування було проведено розгортання системи в робочому середовищі. Наявність статусу Up (running) для всіх компонентів підтверджує коректність налаштування внутрішніх мереж та доступність портів для збору метрик і візуалізації. Результат успішного запуску всіх сервісів представлено на рис. 3.2.

```
root@mon-influxdb-test:/opt/influxv2/nginx# docker-compose ps
  Name                 Command               State             Ports
-------------------------------------------------------------------------------
grafana    /run.sh                          Up      0.0.0.0:3000->3000/tcp
influxdb   /entrypoint.sh influxd           Up      0.0.0.0:8086->8086/tcp
nginx      /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8080->80/tcp
telegraf   /entrypoint.sh telegraf          Up      8092/udp, 8094/tcp, 8125/udp
root@mon-influxdb-test:/opt/influxv2/nginx# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS
   NAMES
daa9466157cf   nginx:alpine                   "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   0.0.0.0:8080->80/tcp
   nginx
dd0e3a9dfb10   telegraf:1.25-alpine           "/entrypoint.sh tele…"   4 minutes ago   Up 3 seconds   8092/udp, 8125/udp, 8094/tcp
   telegraf
996a5888463f   influxdb:2.7-alpine            "/entrypoint.sh infl…"   4 minutes ago   Up 4 minutes   0.0.0.0:8086->8086/tcp
   influxdb
1f4320b771b7   grafana/grafana-enterprise     "/run.sh"                4 minutes ago   Up 4 minutes   0.0.0.0:3000->3000/tcp
   grafana
```

Рисунок 3.2 – Статус активних контейнерів моніторингового стеку

Як видно з результатів виконання команди контролю статусу (рис. 3.2), усі сервіси (Grafana, InfluxDB, Nginx, Telegraf) функціонують стабільно. Використання легковажних образів на базі Alpine Linux дозволило мінімізувати споживання ресурсів хост-системи, забезпечуючи при цьому високу швидкість обробки даних та миттєву готовність до роботи.

3.2. Конфігурування середовища збереження даних InfluxDB

Наступним етапом після запуску контейнерів стала підготовка бази даних InfluxDB OSS 2.x до прийому телеметрії. Оскільки система використовує модель авторизації на основі токенів, основний акцент було зроблено на створенні захищеного каналу передачі даних між збирачем та сховищем.

В межах цього підпункту виконано такі кроки: Налаштування ієрархії доступу. В інтерфейсі керування створено організацію під назвою telegraf та

цільове сховище (bucket) esxi. Саме такий розподіл дозволяє відокремити метрики віртуалізації від інших системних логів.

Генерація API-токена. Для забезпечення зв'язку було випущено унікальний ідентифікатор (Admin Token), який згодом додається у файл telegraf.conf. При виборі параметрів безпеки керуємось офіційними рекомендаціями щодо керування токенами в InfluxDB 2.x [11]. Це дозволило налаштувати доступ за принципом найменших привілеїв, де кожен агент має права лише на запис у свій бакет.

Це дозволило агенту проходити автентифікацію без використання класичних логінів та паролів у відкритому вигляді.

Валідація мережевої взаємодії. Оскільки сервіси працюють у ізольованій мережі Docker, було налаштовано звернення до бази за адресою http://influxdb:8086. Це забезпечує стабільний внутрішній зв'язок незалежно від зовнішньої IP-адреси хоста.

Для контролю за цілісністю даних на початковому етапі скористаємось інструментарієм скриптових запитів. Це дозволить переконатися, що часові ряди записуються з правильною періодичністю (10 секунд, як вказано в налаштуваннях агента). Приклад структури такого запиту, що використовувався для перевірки, винесено у додаток Д.

Така послідовність дій дозволила створити надійне підґрунтя для подальшої візуалізації метрик, виключивши втрати даних при їх передачі з віртуального середовища.

3.3. Конфігурування агента збору даних Telegraf та інтеграція з VMware vSphere

Після підготовки сховища в InfluxDB, переходемо до налаштування системного агента Telegraf. У побудованій архітектурі цей компонент виконує роль інтелектуального шлюзу: він звертається до API VMware vCenter, збирає сирі дані про стан об'єктів та перетворює їх на формат Line Protocol для подальшого аналізу, [35].

Для точного визначення назв лічильників продуктивності та методів звернення до SDK за основу було взято технічну специфікацію Telegraf vSphere Input Plugin [12]. Це дало змогу чітко розмежувати типи даних, які зчитуються з віртуальних машин та фізичних хостів, уникаючи надмірного навантаження на мережу.

3.3.1. Ініціалізація підключення та авторизація джерела даних

Процес побудови зв'язку розпочався у вебінтерфейсі управління InfluxDB. Визначаємо цільовий бакет, та обраємо метод інтеграції, що дозволило системі автоматично згенерувати базові параметри підключення (рис. 3.3).

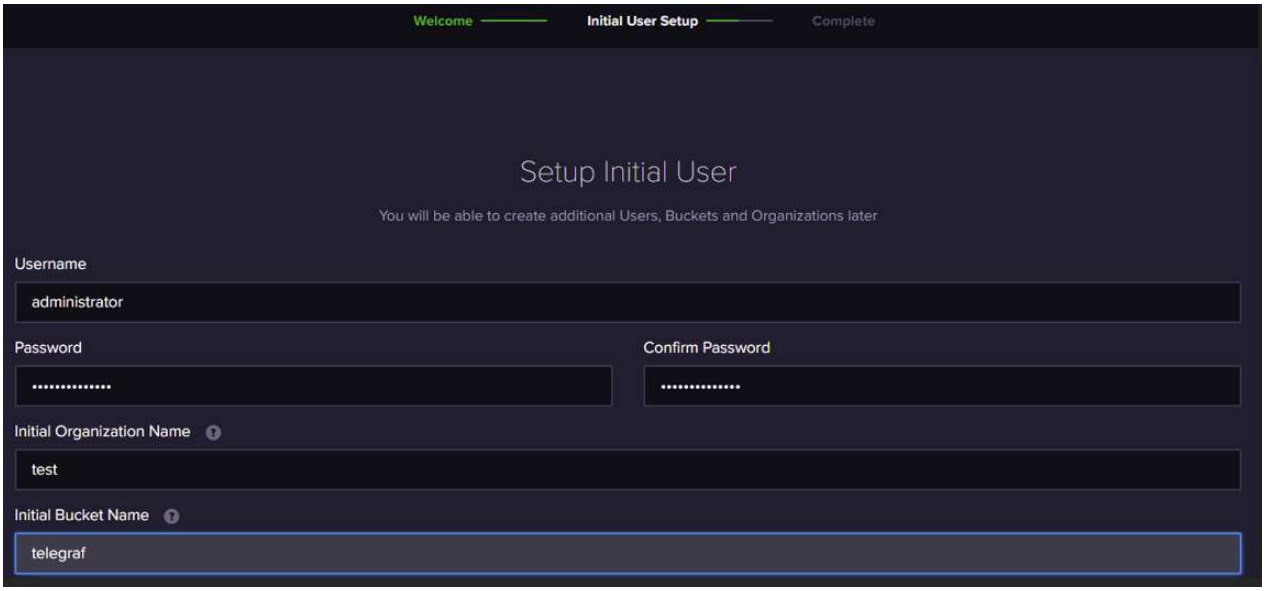


Рисунок 3.3 – Інтерфейс ініціалізації конфігурації Telegraf у середовищі InfluxDB

Як показано на рис. 3.3, під час первинного налаштування доступу для моніторингу ЦОД було визначено наступні параметри:

Username (administrator): основний обліковий запис, який використовуємо для повного контролю та управління правами доступу всередині бази даних.

Initial Organization Name (test): логічне об’єднання ресурсів під назвою «test». У реальних умовах експлуатації це значення можна змінити на офіційну назву підрозділу чи підприємства.

Initial Bucket Name (telegraf): створене як початкове сховище, куди на першому етапі спрямовується потік телеметрії з віртуальних машин та хостів VMware.

Основним кроком після візуального налаштування стала безпосередня правка конфігураційного файлу telegraf.conf. Для забезпечення стабільної взаємодії з віртуальною інфраструктурою задіян модуль [[inputs.vsphere]]. Параметри доступу до vCenter, згідно з робочою конфігурацією, наведено у лістингу 3.2, а повний текст налаштованого файлу представлено у додатку Б.

Під час налагодження звертаємо увагу на параметр insecure_skip_verify = true. Свідомо залишаємо його активованим, що дозволило агенту успішно збирати дані без необхідності імпорту SSL-сертифікатів, що значно прискорило процес тестування у локальному середовищі.

Наступним критично важливим етапом є забезпечення безпечного доступу. У моделі InfluxDB v2 автентифікація відбувається за допомогою API-токенів, [4]. На рис. 3.4 зафіксовано завершення етапу створення системного токена.

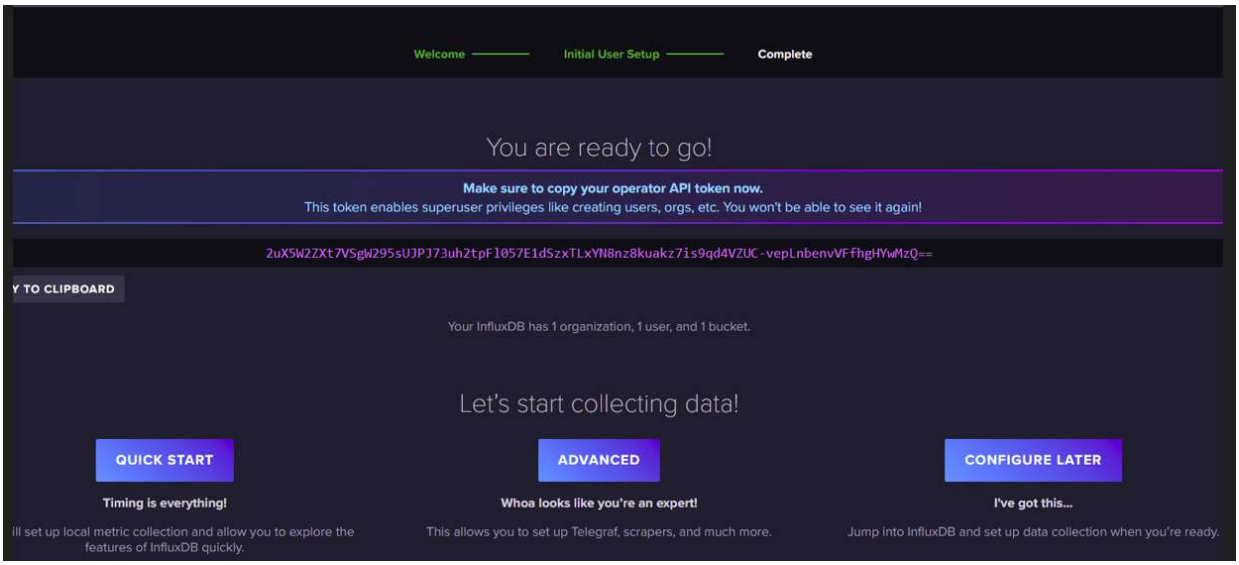


Рисунок 3.4 – Підтвердження успішної генерації API-токена

Згенерований «Operator Token» (рис. 3.4) має повні права на читання та запис у межах створеної організації. Саме цей ідентифікатор у подальшому використовується в конфігураційних файлах Telegraf та Grafana для легітимного доступу до даних. Такий підхід дозволяє уникнути передачі відкритих паролів мережею, що значно підвищує рівень кібербезпеки системи управління ЦОД.

Важливим етапом налаштування бази даних є створення логічних сховищ – бакетів (Buckets). Для системи моніторингу ЦОД необхідно не лише визначити назву сховища, а й чітко встановити параметри життєвого циклу даних.

Процес ініціалізації нового бакета представлено на рис. 3.5.

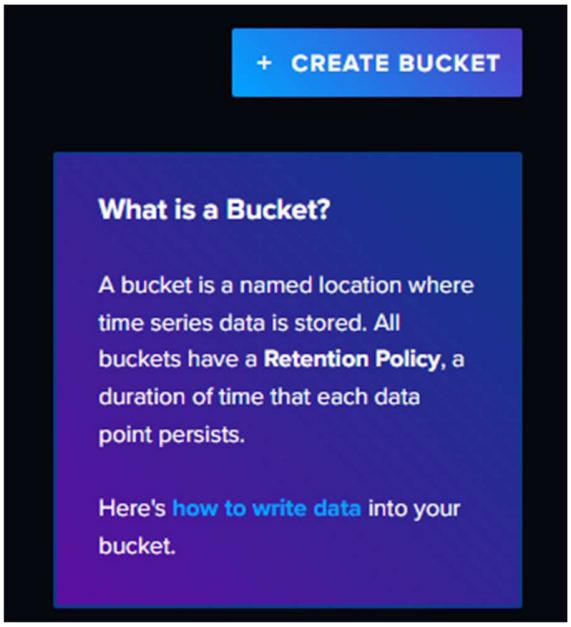


Рисунок 3.5 – Інтерфейс створення цільового бакета в InfluxDB

Як показано на рис. 3.2, сховищу було присвоєно назву telegraf, що відповідає назві агента збору даних. Це спрощує ідентифікацію джерела телеметрії при подальшому масштабуванні системи.

Особлива увага приділяється параметру «Delete Data», який відповідає за політику автоматичного очищення застарілих записів. На рис. 3.6 продемонстровано вибір терміну утримання даних.

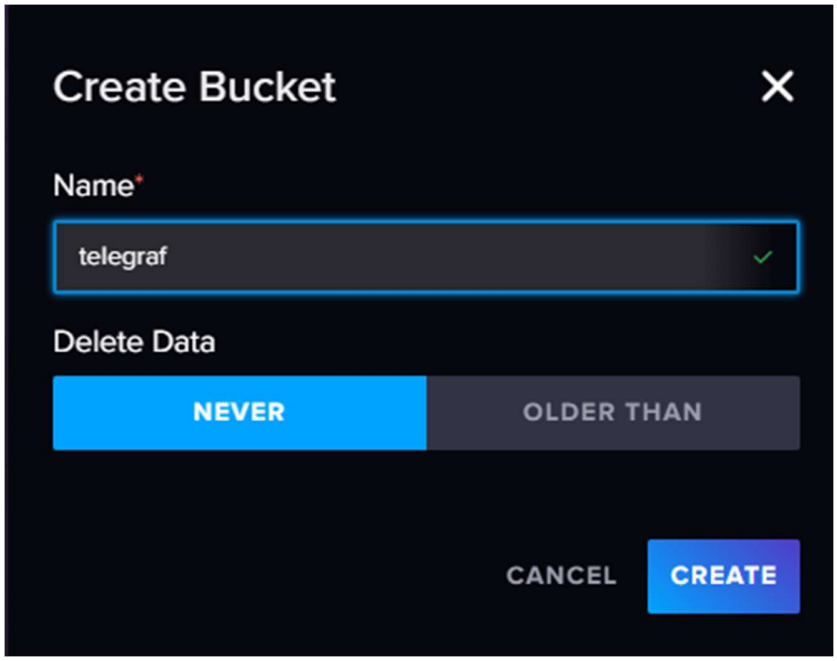


Рисунок 3.6 – Налаштування Retention Policy для бакета telegraf

Для забезпечення максимальної глибини ретроспективного аналізу та можливості побудови довгострокових трендів продуктивності інфраструктури, було обрано політику "Never" (рис. 3.6). Таке рішення дозволяє зберігати повну історію метрик без автоматичного видалення, що є критично важливим для проведення аудиту інцидентів та планування потужностей (capacity planning) ЦОД у довгостроковій перспективі.

Фінальна верифікація створеної структури відображена у загальному списку активних сховищ (рис. 3.7).

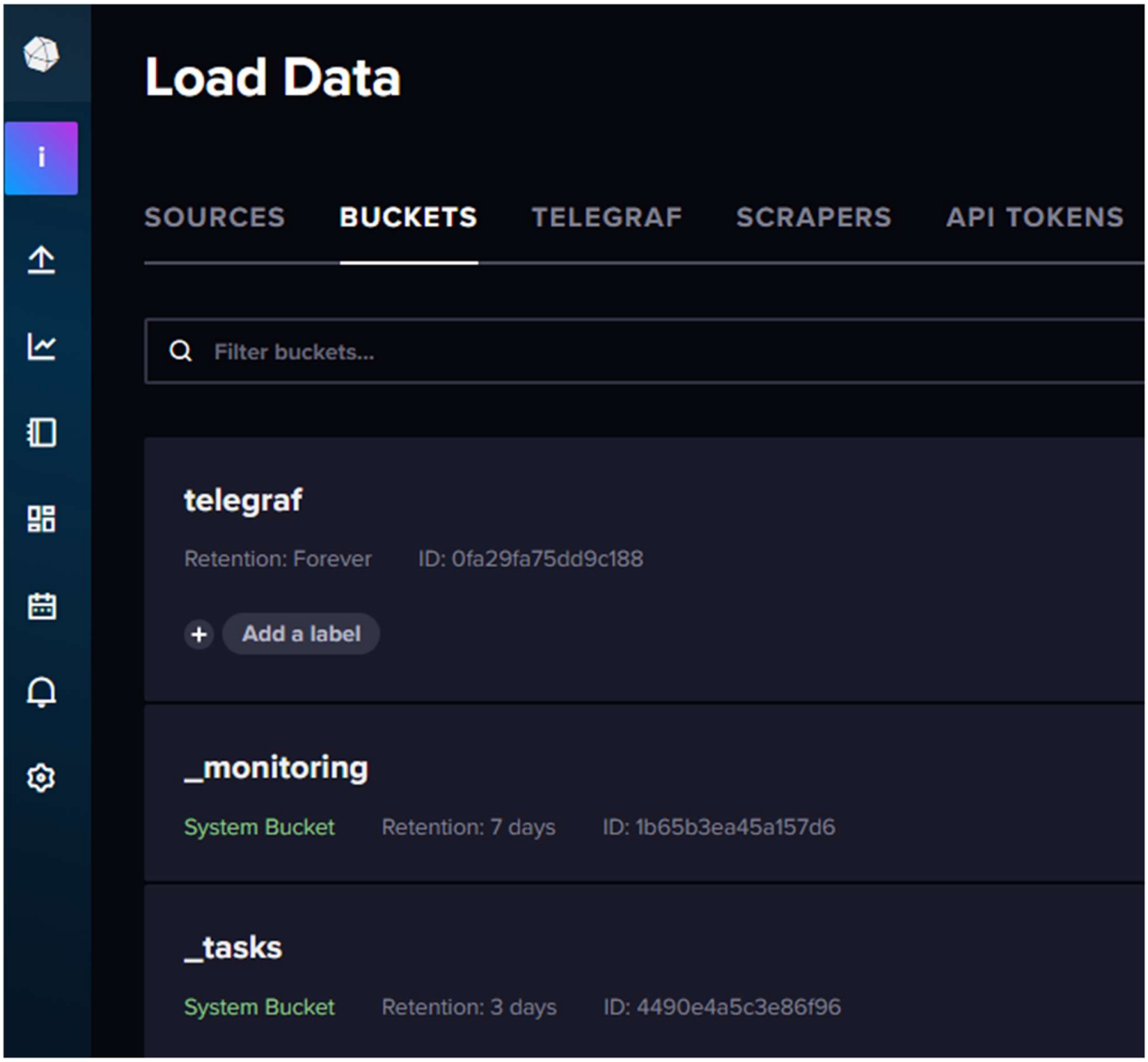


Рисунок 3.7 – Підтвердження успішного створення бакета з встановленою політикою зберігання

Наявність у списку бакета telegraf із міткою «Forever» (рис. 3.7) підтверджує повну готовність бази даних до прийому часових рядів. На цьому етап підготовки середовища зберігання вважається завершеним.

Основним кроком є редагування файлу telegraf.conf. Для забезпечення зв'язку з віртуальною інфраструктурою використовується плагін [[inputs.vsphere]]. Параметри підключення до vCenter, згідно з робочою

конфігурацією, наведено у лістингу 3.2, а повний текст налаштованого файлу з усіма активованими модулями збору даних міститься у додатку Б.

Використання облікового запису “Ім’я-read-only” (read-only) гарантує безпеку, оскільки агент має права лише на читання метрик, без можливості зміни конфігурації серверів.

1. Ініціалізація підключення джерела даних. Процес налаштування розпочинається у вебінтерфейсі управління сервісами, де здійснюється вибір цільового бакета та методу збору даних (рис. 3.8).

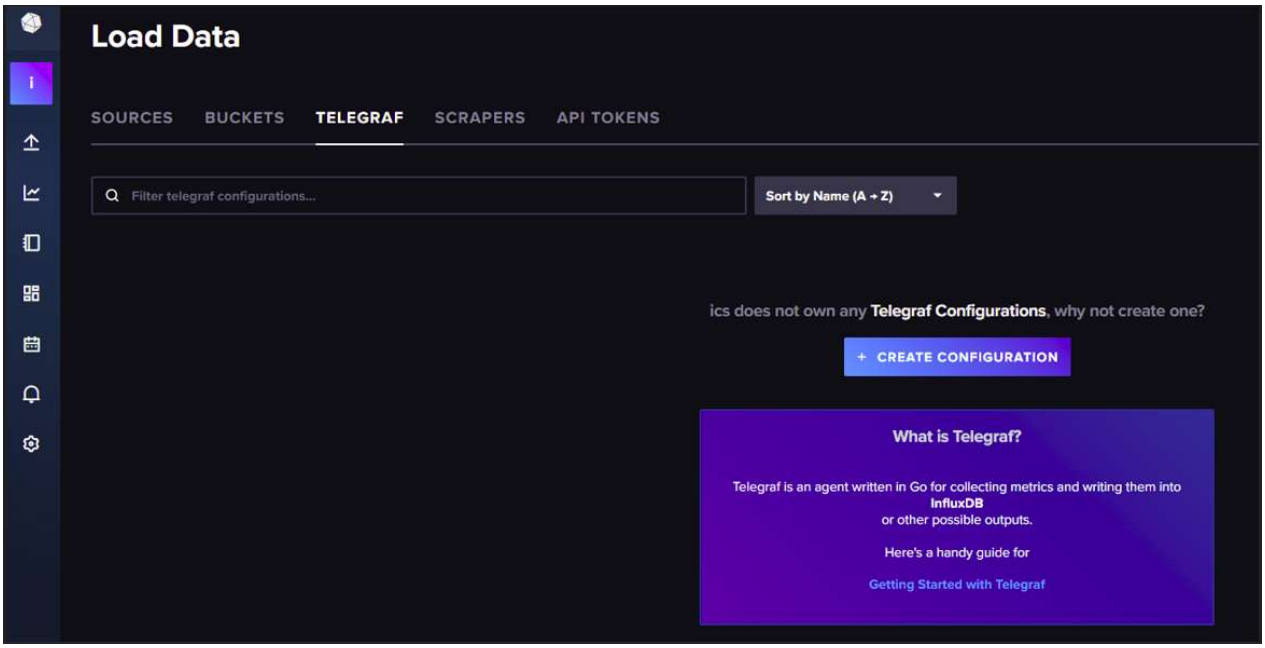


Рисунок 3.8 – Вибір бакета та інструментів для збору телеметрії

Як показано на рис. 3.8, для інтеграції обрано раніше створене сховище telegraf. Використання вбудованих майстрів налаштування InfluxDB дозволяє згенерувати базовий шаблон конфігурації, який у подальшому адаптується під специфічні потреби інфраструктури віртуалізації.

2. Налаштування плагіна inputs.vsphere. Основним інструментом збору даних є плагін inputs.vsphere, який використовує SDK для взаємодії з vCenter Server.

Для забезпечення збору телеметрії з гіпервізорів та віртуальних машин було обрано та активовано спеціалізований вхідний плагін (Input Plugin) – vsphere. Процес його ініціалізації в середовиці конфігуратора представлено на рис. 3.9.

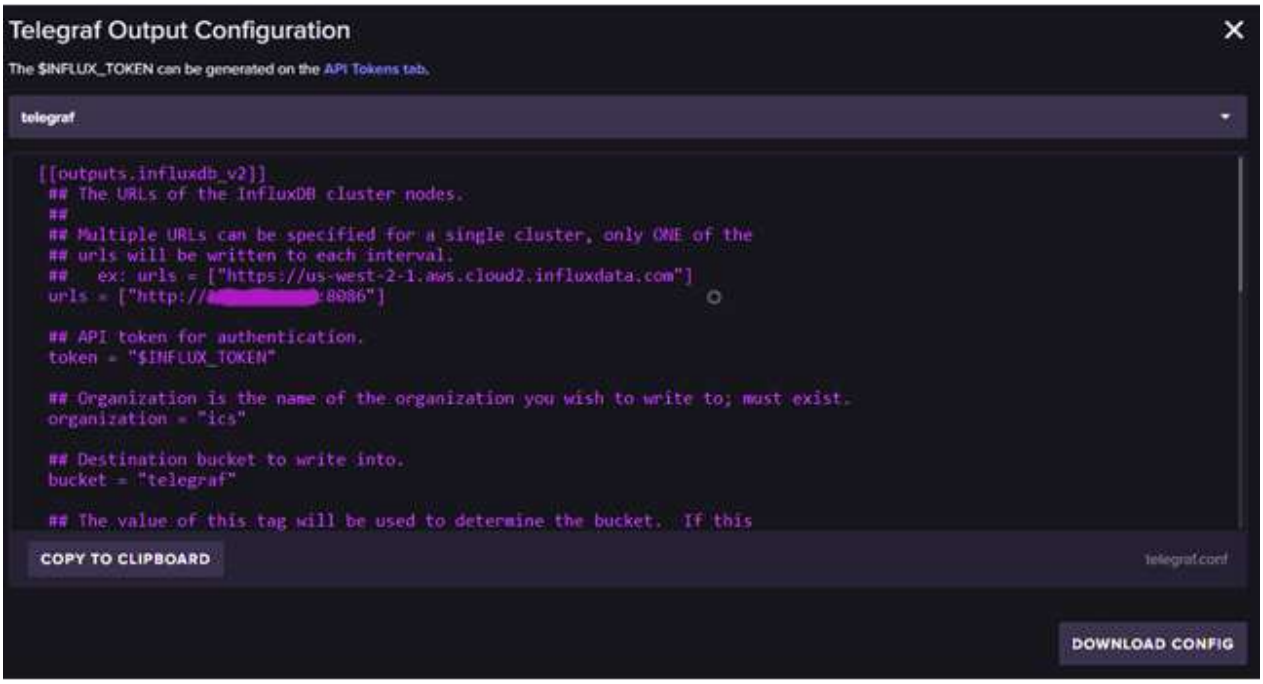


Рисунок 3.9 – Вибір та ініціалізація вхідного плагіна vSphere

Як показано на рис. 3.9, даний плагін є ключовим компонентом архітектури моніторингу. Він дозволяє агенту Telegraf взаємодіяти безпосередньо з API vCenter Server. Використання цього модуля забезпечує:

Уніфікацію даних: перетворення метрик VMware у формат, зрозумілий для бази даних InfluxDB.

Гнучкість: можливість детального налаштування інтервалів опитування та категорій об'єктів (обчислювальні ресурси, мережі, сховища) на наступних етапах конфігурування.

Цей крок є фундаментом для створення безперервного потоку даних про стан IT-інфраструктури підприємства.

Після активації плагіна, необхідно налаштувати параметри автентифікації, для доступу до API віртуальної інфраструктури. На рис. 3.9 представлено інтерфейс введення облікових даних.

3.3.2. Налаштування вихідного потоку даних (Output Plugin)

Останнім етапом конфігурування є спрямування зібраних даних до InfluxDB 2.x. Згідно з наданим файлом конфігурації, передача здійснюється через плагін outputs.influxdb_v2.

Лістинг 3.3 – Фрагмент конфігурації вихідного плагіна для передачі даних

```
[[outputs.influxdb_v2]]
  urls = ["http://influxdb:8086"]
  token = "nWW3sw...CvaqQQ=="
  organization = "telegraf"
  bucket = "esxi"
```

Завдяки тому, що Telegraf та InfluxDB знаходяться в одній мережі Docker (monitoring), передача даних відбувається за внутрішнім іменем хоста http://influxdb, що забезпечує стабільність зв'язку при зміні зовнішніх IP-адрес вузла.

4. Гранулярне налаштування об'єктів моніторингу та метрик

Після визначення загальних параметрів підключення необхідно провести тонке налаштування збору телеметрії. Це дозволяє уникнути надлишковості даних та зосередитися на критично важливих показниках продуктивності. На рис. 3.10 представлено програмну реалізацію цих фільтрів у конфігурації Telegraf.

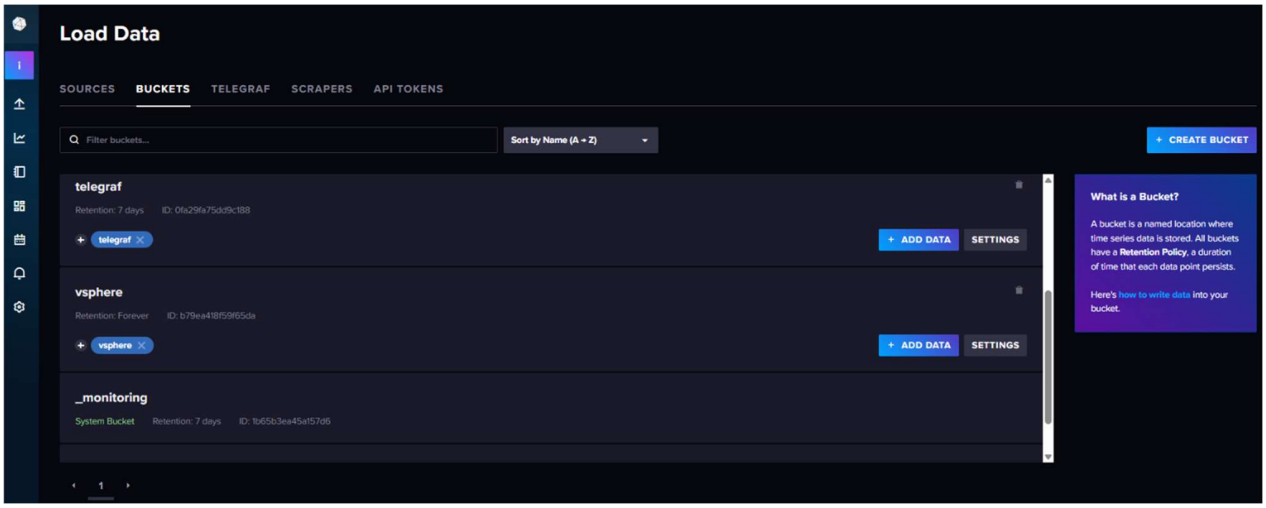


Рисунок 3.10 – Програмне налаштування фільтрації ресурсів та вибору метрик у плагіні vSphere

Як показано на рис. 3.10, конфігурація включає деталізований перелік параметрів, що підлягають збору. Зокрема, для дискової підсистеми (Datastores) було визначено такі ключові метрики:

Capacity ma Used: для оперативного контролю за заповненням сховищ та запобігання зупинкам віртуальних машин через брак місця.

Read/Write Latency: для моніторингу затримок під час операцій введення-виведення, що є основним показником "здоров'я" систем зберігання даних.

Datastore Instances: активовано параметр true, що дозволяє отримувати статистику в розрізі кожного окремого тому, а не лише загальні показники по кластеру.

Використання такого підходу (рис. 3.10) гарантує, що адміністратор отримає вичерпну інформацію про роботу ЦОД, при цьому навантаження на мережу та базу даних залишиться в межах норми завдяки відсіканню неінформативних параметрів.

5. Верифікація працездатності каналу збору телеметрії

Завершальним кроком у налаштуванні агента є перевірка фактичного надходження даних у сховище. Для цього використано інструмент аналізу InfluxDB – Data Explorer (рис. 3.11).

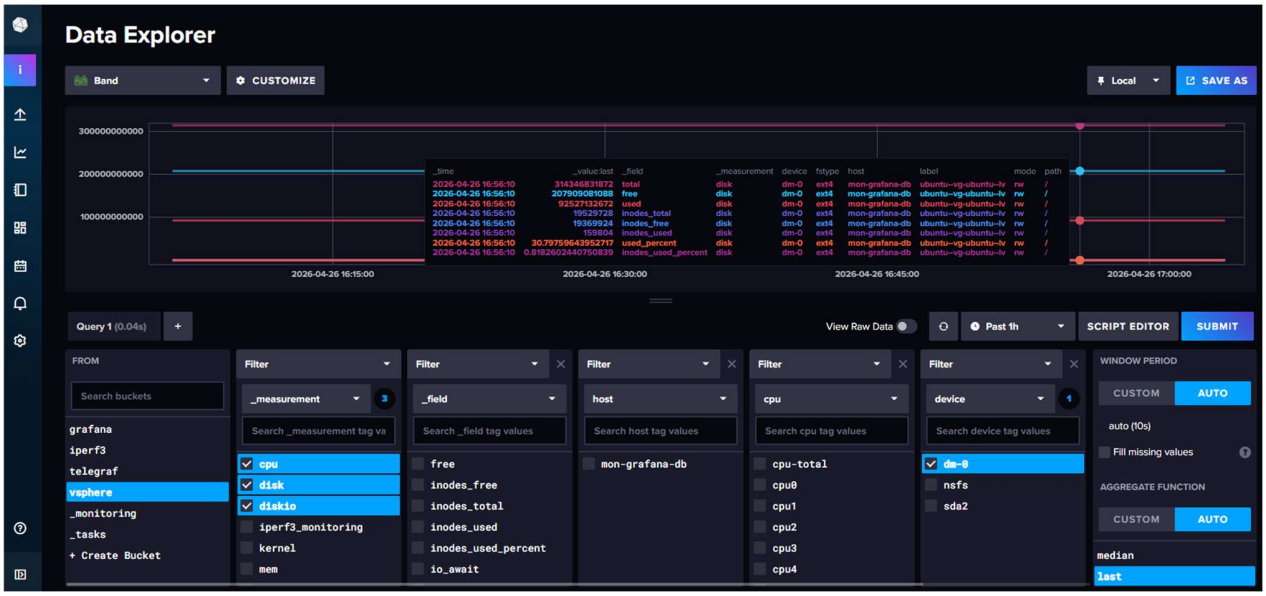


Рисунок 3.11 – Візуальна перевірка отриманих часових рядів у інтерфейсі Data Explorer

Наявність структурованих графіків на рис. 3.11 підтверджує, що весь ланцюжок моніторингу – від vCenter API до бази даних InfluxDB – працює коректно. Кожна точка на графіку відповідає реальному стану інфраструктури в конкретний момент часу, що дозволяє переходити до фінального етапу проекту – створення аналітичних панелей.

3.4 Візуалізація метрик та побудова аналітичних панелей у Grafana

Останнім етапом розгортання системи моніторингу є налаштування платформи Grafana. Її роль полягає у перетворенні сирих даних часових рядів, що зберігаються в InfluxDB, у динамічні інтерактивні дашборди. [6]. Це дозволяє персоналу ЦОД оперативно реагувати на аномалії та візуально оцінювати стан інфраструктури віртуалізації.

1. Інтеграція Grafana з базою даних InfluxDB

Процес розпочинається з додавання нового джерела даних (Data Source). На рис. 3.12 представлено конфігурацію підключення до InfluxDB.

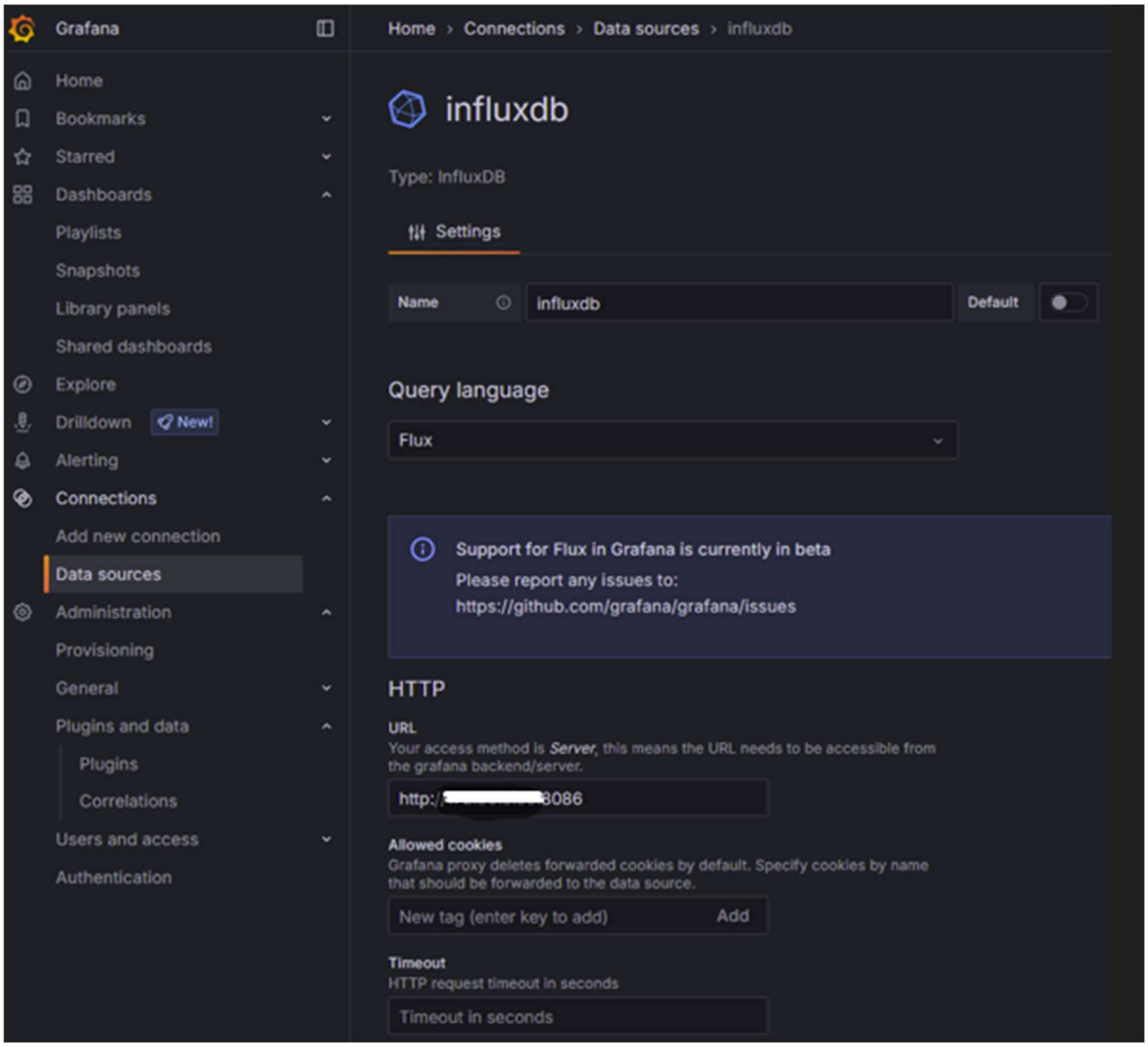


Рисунок 3.12 – Конфігурування InfluxDB як джерела даних у Grafana

При виконанні налаштувань, відображених на рис. 3.12, було враховано такі архітектурні особливості:

Query Language: обрано мову запитів Flux. На відміну від класичного InfluxQL, Flux дозволяє виконувати складну математичну обробку даних безпосередньо на стороні бази, що критично важливо для обчислення відсоткових показників завантаження CPU та RAM у реальному часі.

HTTP Connection: вказано мережеву адресу сервера http://ip-address:8086. Використання прямого IP-підключення забезпечує мінімальні затримки при передачі пакетів телеметрії між контейнерами моніторингу.

Access Mode: встановлено режим Server, що означає виконання запитів від імені бекенда Grafana. Це підвищує рівень безпеки, оскільки токени доступу не передаються у браузер клієнта.

1.2. Верифікація авторизації та доступу

Для забезпечення доступу до конкретних часових рядів, необхідно вказати деталі організації, та API-токен. На рис. 3.13 продемонстровано введення параметрів ідентифікації.

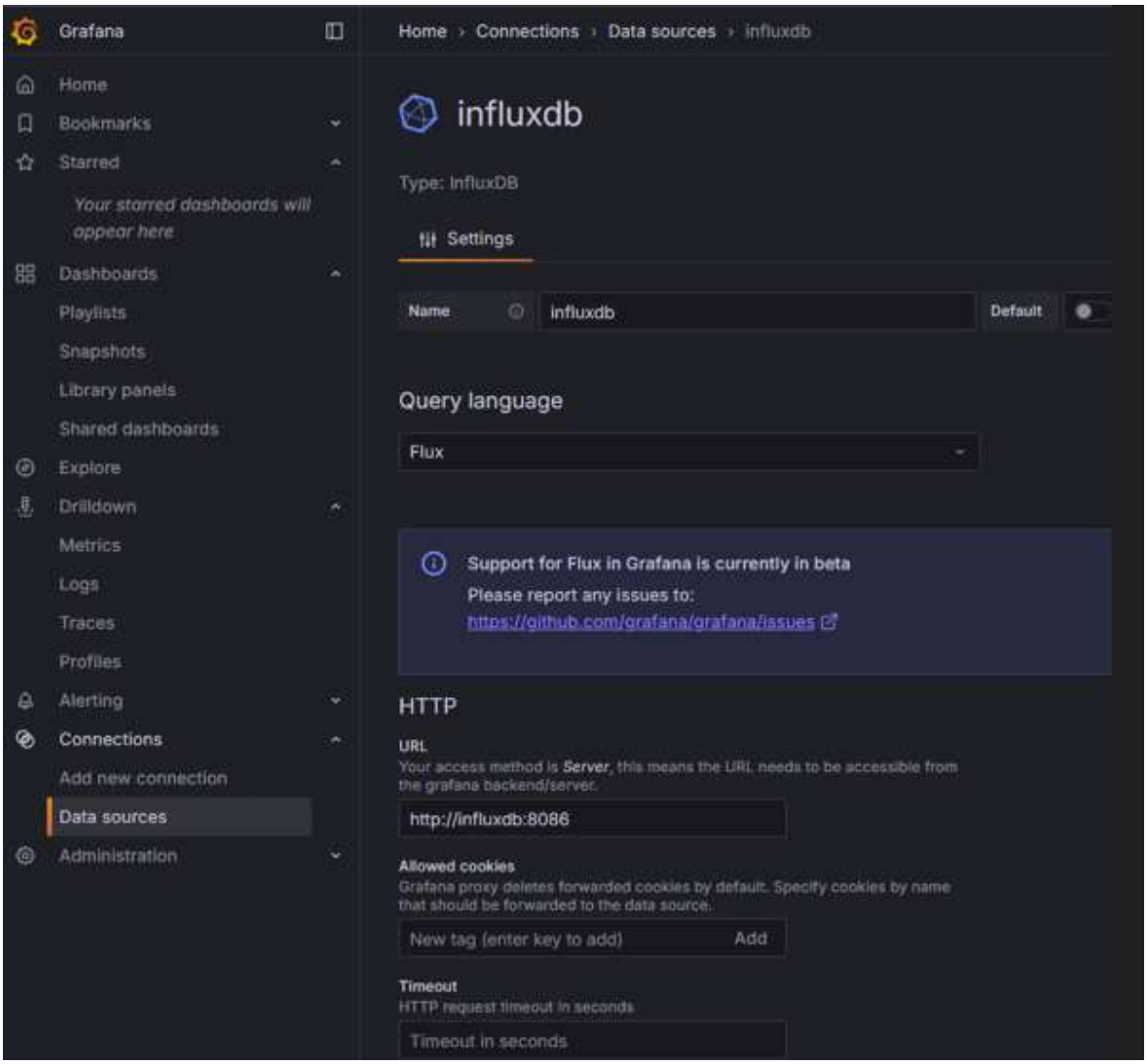


Рисунок 3.13 – Налаштування автентифікації та вибір цільової організації

Як показано на рис. 3.13, Grafana підключається до організації, що була створена на етапі ініціалізації InfluxDB. Це гарантує ізоляцію даних моніторингу ЦОД від інших можливих проектів усередині бази. Після введення токена, виконується перевірка зв’язку, яка підтверджує готовність системи до побудови графічних панелей.

1.3. Верифікація успішності з’єднання

Фінальним етапом підключення джерела, є перевірка зв’язку між сервісами. Результат натискання кнопки «Save & test» зафіксовано на рис. 3.14.

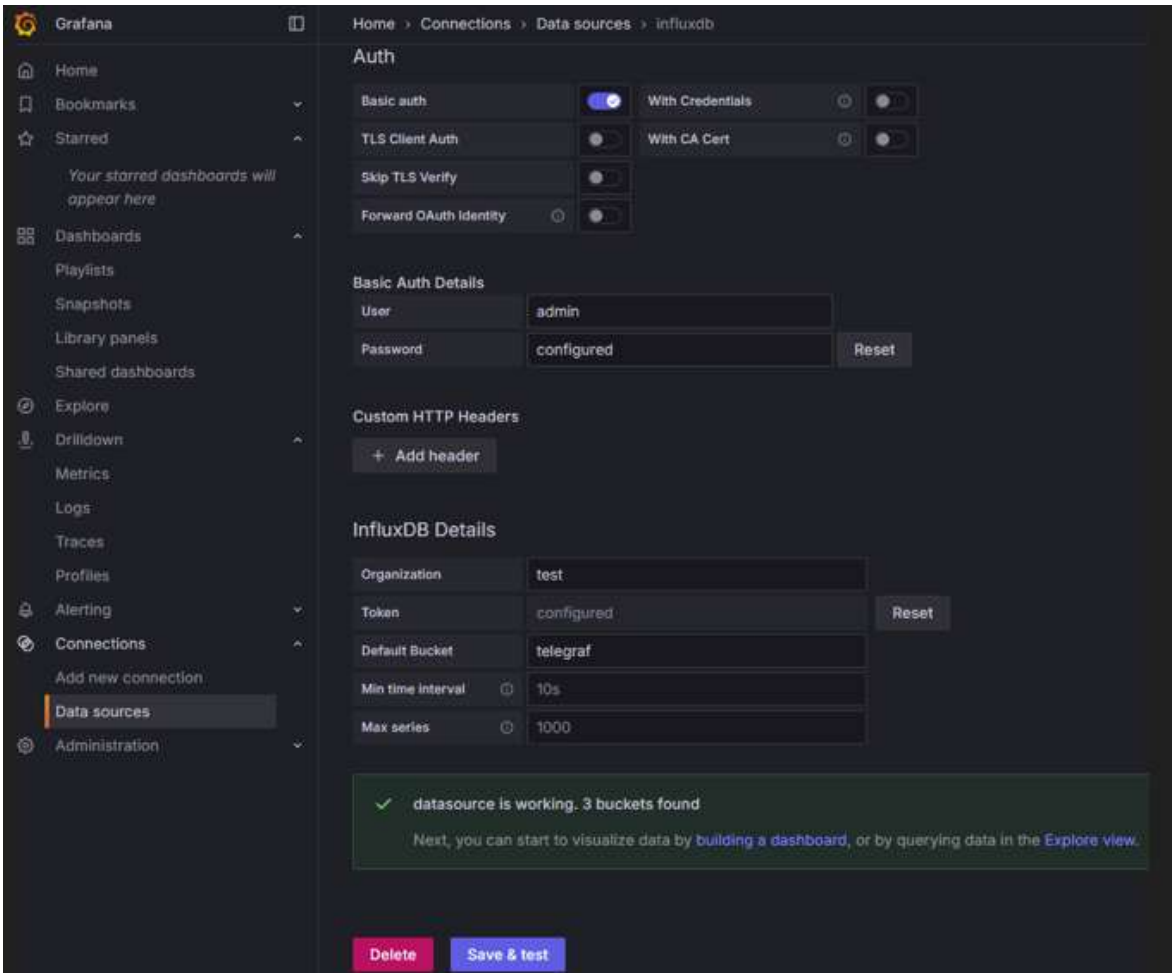


Рисунок 3.14 – Підтвердження успішної авторизації («datasource is working»)

Повідомлення на рис. 3.14 підтверджує, що Grafana успішно розпізнала структуру бази даних, (знайдено 5 бакетів). Це свідчить про повну працездатність стеку моніторингу та дозволяє перейти до формування графічних інтерфейсів.

3.5 Розробка та налаштування графічних панелей моніторингу

Після підключення джерела даних, було проведено роботу з проектування дашбордів, які дозволяють у наочній формі відстежувати стан ЦОД.

Для візуалізації стану інфраструктури ЦОД, було обрано шлях інтеграції професійних шаблонів з репозиторію Grafana Labs. [21] Це дозволяє використовувати перевірені часом методики відображення складних метрик vSphere.

1. Імпорт спеціалізованого дашборда за ID Основним інструментом візуалізації було обрано дашборд з ідентифікатором 8159, [22] який є стандартом де-факто для моніторингу середовищ VMware. Процес ініціалізації представлено на рис. 3.15.

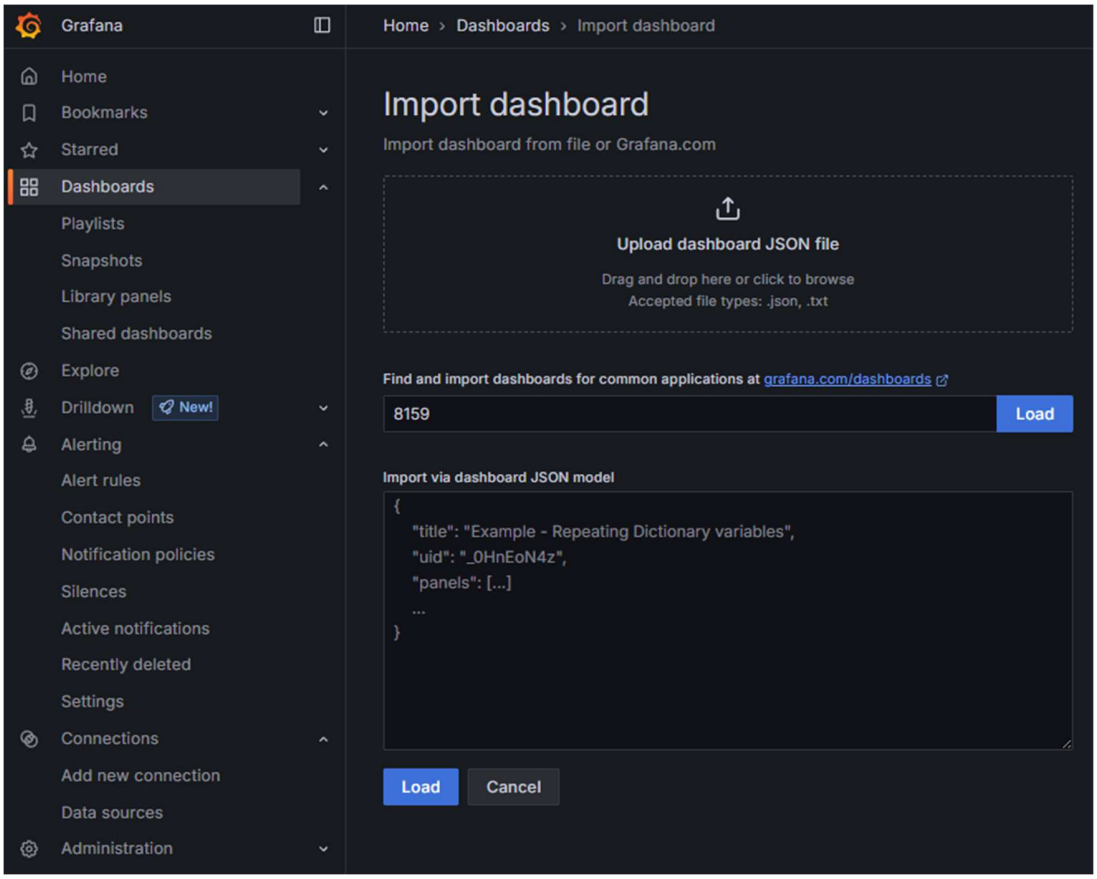


Рисунок 3.15 – Використання механізму імпорту Grafana для розгортання дашборда №8159

Як показано на рис. 3.15, система дозволяє завантажити готову структуру аналітичних панелей безпосередньо з офіційного сайту. Це включає автоматичне створення десятків графіків для обчислювальних ресурсів, мереж та систем зберігання даних.

1.2. Налаштування параметрів відображення та вибір джерела даних Після завантаження структури шаблону необхідно провести фінальну прив’язку до локального середовища. На рис. 3.16 представлено етап вибору джерела даних.

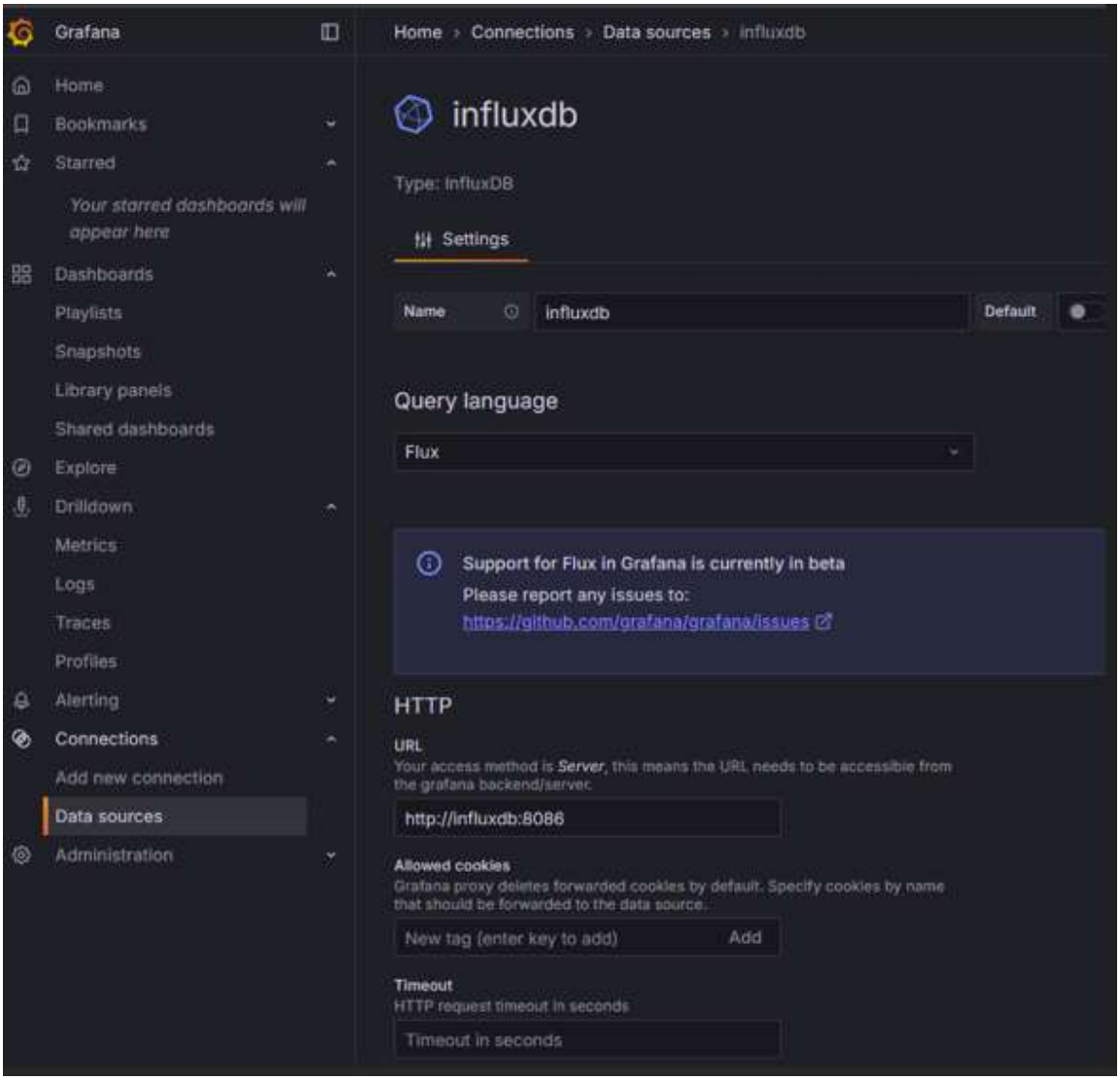


Рисунок 3.16 – Фінальне налаштування та вибір InfluxDB для імпортованої панелі

Як показано на рис. 3.16, на цьому кроці адміністратор вказує, з якого саме бакета (у нашому випадку telegraf) дашборд має зчитувати метрики. Це забезпечує автоматичне заповнення всіх графіків реальними даними без необхідності ручного редагування кожного окремого віджета.

3.6. Структура збору та мапінгу метрик інфраструктури

Для забезпечення точності візуалізації та коректності роботи тригерів було розроблено схему відповідності між системними показниками VMware vSphere та полями в базі даних InfluxDB. Основні метрики, що використовуються в роботі, представлені у таблиці 3.1. [3, 6, 7].

Таблиця 3.1 – Відповідність метрик vSphere полям у базі даних InfluxDB

Об'єкт моніторингу	**Метрика vSphere**	**Поле (Field) в InfluxDB**	**Одиниця виміру**	**Призначення**
Virtual Machine	cpu.usage.average	cpu_usage_average	%	Контроль загального навантаження на процесор
Virtual Machine	mem.usage.average	mem_usage_average	%	Моніторинг споживання оперативної пам'яті
Virtual Machine	cpu.ready.summation	cpu_ready_summation	ms	Виявлення черг на фізичний процесор
Datastore	disk.capacity.latest	capacity_latest	GB	Аналіз загального об'єму сховища
Datastore	disk.used.latest	used_latest	GB	Відстеження динаміки заповнення дисків
Host (ESXi)	net.usage.average	net_usage_average	KBps	Контроль пропускної здатності мережі

3.7 Аналіз результатів та експлуатація системи моніторингу

Фінальним етапом роботи стало розгортання та запуск інтерактивної панелі моніторингу. Після завершення процедури імпорту (рис. 3.15) та підтвердження зв'язку з базою даних, система автоматично сформувала багаторівневий інтерфейс візуалізації. [32]. Результат представлено на рис. 3.17.



Рисунок 3.17 – Підсумковий дашборд моніторингу інфраструктури vSphere у Grafana

Відповідно до рис. 3.17, розроблена система забезпечує контроль над наступними критичними зонами інфраструктури:

Cluster & Infrastructure Summary: Верхній рядок панелей надає кількісну інформацію про об’єкти (1 vCenter, 3 кластери, 5 ESXi-хостів та 33 віртуальні машини). Це дозволяє миттєво оцінити масштаб контрольованої мережі.

Resource Overview (CPU/RAM): Кольорові індикатори («tiles») демонструють відсоткове завантаження ресурсів для різних сегментів (наприклад, AMD-LON, CL-LON). Зелений колір індикаторів (завантаження CPU

20.60% та 22.05%) свідчить про стабільну роботу та наявність вільного ресурсу для масштабування.

Datastores Usage Capacity: Візуалізація заповненості сховищ (від 3% до 84%) дозволяє адміністратору прогнозувати вичерпання дискового простору та вчасно проводити очищення або розширення LUN.

Hypervisor & VM Status: Графіки часових рядів відображають динаміку споживання ресурсів кожним окремим вузлом та віртуальною машиною, що дає змогу виявляти «пікові» навантаження та локалізувати проблемні сервіси.

Disk Performance (Latency & IOPS): Нижня частина панелі (рис. 3.18) фокусується на продуктивності дискової підсистеми. Моніторинг затримок (Latency) є критично важливим для баз даних та високонавантажених застосунків.

Для моніторингу дискових підсистем недостатньо знати лише абсолютні значення вільного об'єму. Для отримання відносних показників завантаженості СЗД у реальному часі було розроблено запит, що об’єднує потоки даних capacity_latest та free_space_latest. Математична модель цього розрахунку з використанням функції join представлена у Додатку Г.

3.8 Спеціалізовані панелі моніторингу компонентів інфраструктури

Для забезпечення комплексного підходу до адміністрування було проведено налаштування спеціалізованих дашбордів. Це дозволяє деталізувати стан окремих технологічних стеків ЦОД, не обмежуючись загальними показниками. На рис. 3.18 представлено перелік інстальованих панелей, що інтегровані в єдине середовище Grafana.



Рисунок 3.18 – Структура інтегрованих дашбордів для поглибленого моніторингу

Використання такої ієрархії (рис. 3.18) дозволяє технічному персоналу здійснювати швидкий перехід від глобального огляду інфраструктури до детального аналізу конкретного вузла або віртуального сервера. Далі розглянемо ключові компоненти цієї системи за категоріями.

3.8.1 Моніторинг загального стану кластера та дискових масивів

Першим етапом поглибленого аналізу є використання панелі VMware vSphere – Overview, [24] яка дозволяє одночасно відстежувати обчислювальні потужності та стан систем зберігання. На рис. 3.18 представлено візуалізацію поточного навантаження.

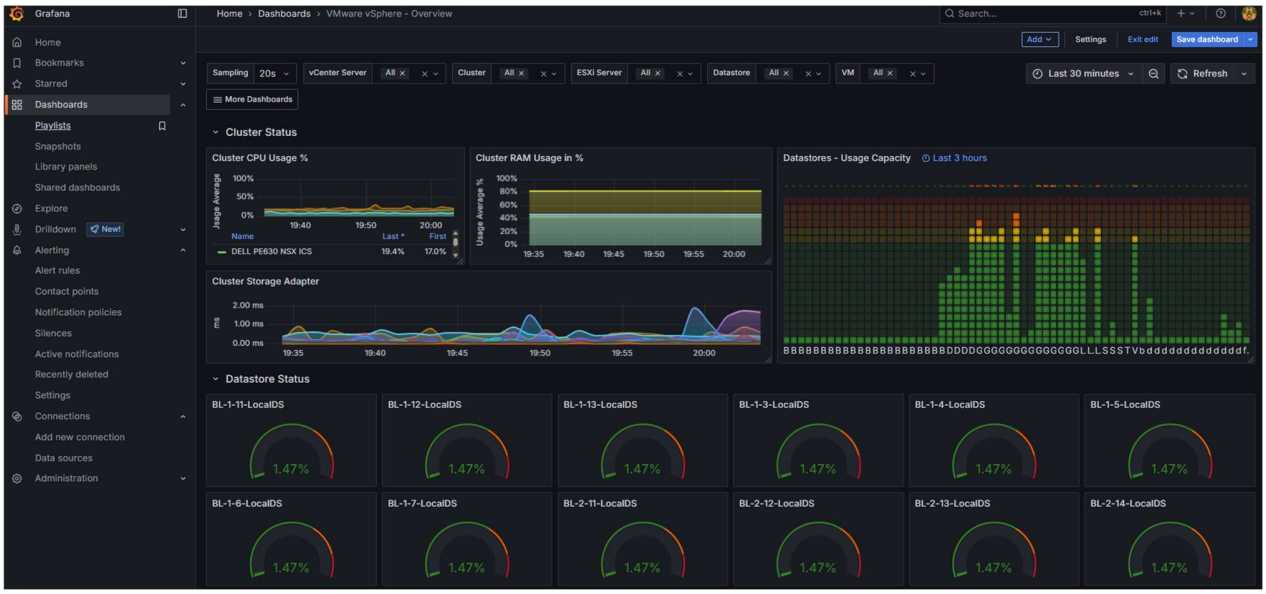


Рисунок 3.18 – Загальний огляд ресурсів кластера та стану локальних сховищ

Відповідно до рис. 3.18, дашборд забезпечує контроль за наступними параметрами:

Cluster CPU & RAM Usage: графіки відображають стабільне споживання ресурсів (близько 19.4% для CPU), що підтверджує відсутність перевантажень у пікові моменти.

Datastores – Usage Capacity: використання «теплової карти» (heatmap) дозволяє візуально ідентифікувати найбільш завантажені дискові масиви. Зелений колір більшості комірок свідчить про достатній запас вільного місця.

Локальні індикатори сховищ (Gauges): Нижня частина панелі (наприклад, BL-1-11-LocalDS та інші) демонструє відсоткове заповнення кожного окремого тому. Показник у 1.47% свідчить про початкову стадію експлуатації або високу ефективність розподілу даних.

Cluster Storage Adapter: моніторинг затримок (latency) у мілісекундах дозволяє вчасно виявити деградацію продуктивності дискової підсистеми на рівні адаптерів хостів.

3.8.2 Контроль продуктивності віртуальних машин

Для детального моніторингу окремих вузлів віртуалізації було налаштовано дашборд VMware vSphere - VMs. [25]. Ця панель дозволяє адміністратору проводити аудит кожної віртуальної машини окремо, аналізуючи ефективність використання виділених ресурсів. Візуалізацію стану конкретних ВМ наведено на рис. 3.20.

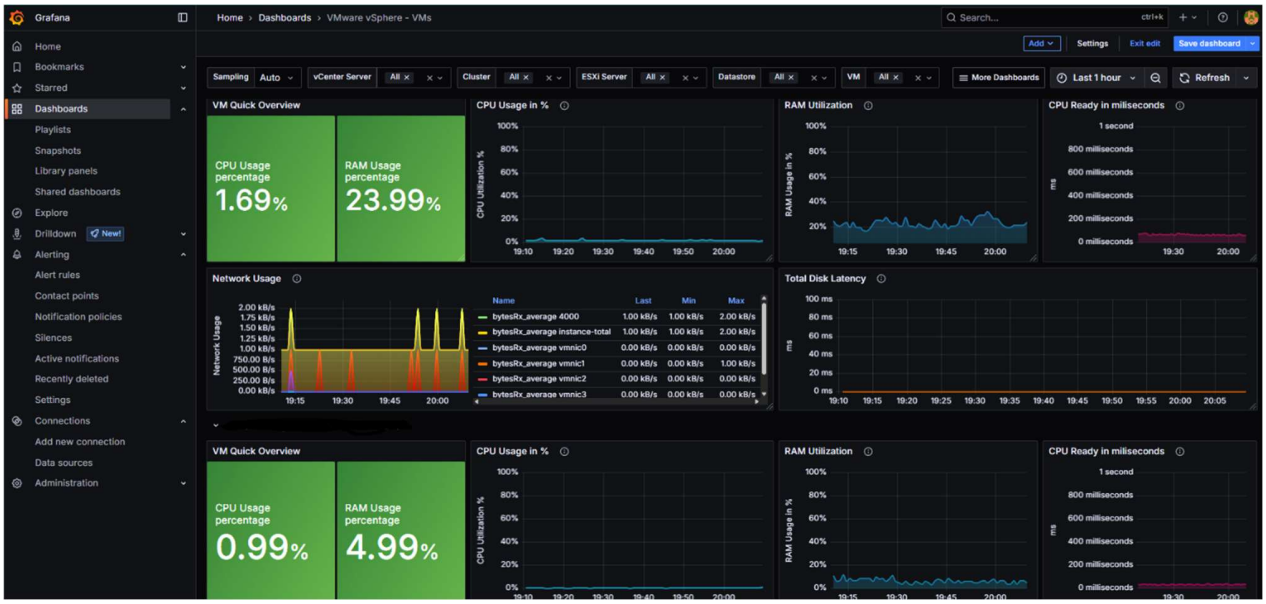


Рисунок 3.20 – Детальний моніторинг ресурсів на рівні окремих віртуальних машин

Як продемонстровано на рис. 3.20, система надає вичерпну інформацію за наступними напрямками:

VM Quick Overview: Компактні плитки зі статистикою (CPU Usage percentage та RAM Usage percentage) дозволяють миттєво ідентифікувати машини, що потребують збільшення ресурсів або мають надлишкове резервування. Наприклад, завантаження у 1.69% та 0.99% CPU свідчить про значний запас потужності.

Network Usage: Графіки мережевої активності відображають трафік у розрізі інтерфейсів (vmnic), що дозволяє виявляти мережеві аномалії або перевантаження каналів зв'язку.

CPU Ready in milliseconds: Це один із найважливіших показників у віртуалізації. Він відображає час, протягом якого ВМ готова до виконання завдань, але очікує звільнення фізичного процесора. Низькі значення на графіку (рис. 3.19) підтверджують відсутність конфліктів за процесорний час на хостах.

Total Disk Latency: моніторинг затримок дискових операцій безпосередньо для кожної ВМ дозволяє локалізувати проблеми продуктивності, пов'язані з конкретними застосунками або базами даних.

3.8.3 Глибокий аналіз систем зберігання даних

Завершальним компонентом системи є дашборд VMware vSphere - Datastore, [22], який фокусується виключно на метриках надійності та місткості дискових ресурсів. На рис. 3.21 представлено детальну візуалізацію стану одного з локальних сховищ.

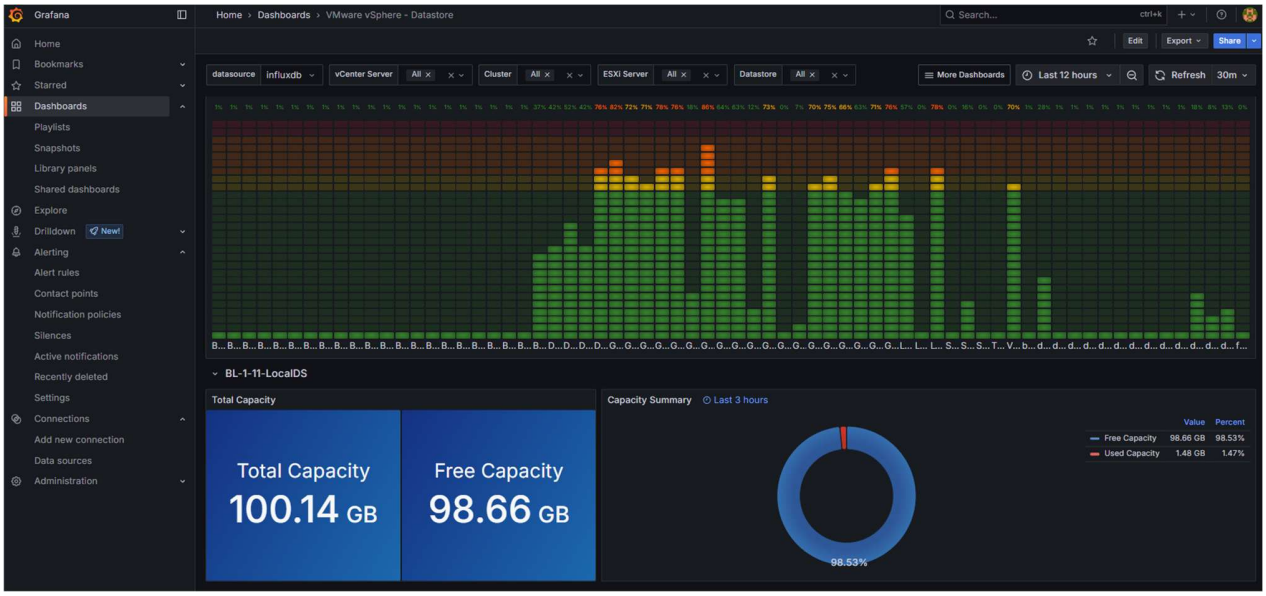


Рисунок 3.21 – Моніторинг ємності та ефективності використання сховищ (Datastores)

Відповідно до даних на рис. 3.21, панель дозволяє виконувати прецизійний контроль наступних показників:

Capacity Heatmap: верхня частина панелі відображає розподіл завантаженості всіх підключених сховищ. Використання градієнтної шкали (від зеленого до помаранчевого) дозволяє адміністратору миттєво локалізувати томи,

де вільне місце наближається до критичного порогу (наприклад, показники 82% та 84% на графіку).

Total & Free Capacity: система надає точні цифрові значення об’єму сховищ. Як видно на прикладі BL-1-11-LocalDS, загальна ємність складає 100.14 GB, з яких вільними залишаються 98.66 GB.

Capacity Summary: кільцева діаграма візуалізує співвідношення використаного та вільного простору у відсотках. Показник 98.53% вільного місця (Free Capacity) свідчить про високу доступність ресурсів для розгортання нових віртуальних машин або розширення існуючих дисків.

Used Capacity Trend: індикатор використання у 1.47% (1.48 GB) підтверджує мінімальне поточне навантаження на обраний локальний диск, що є оптимальним для стабільної роботи системних логів та файлів конфігурації.

На рис. 3.22 представлено візуалізацію стану мережевих сховищ великої місткості.

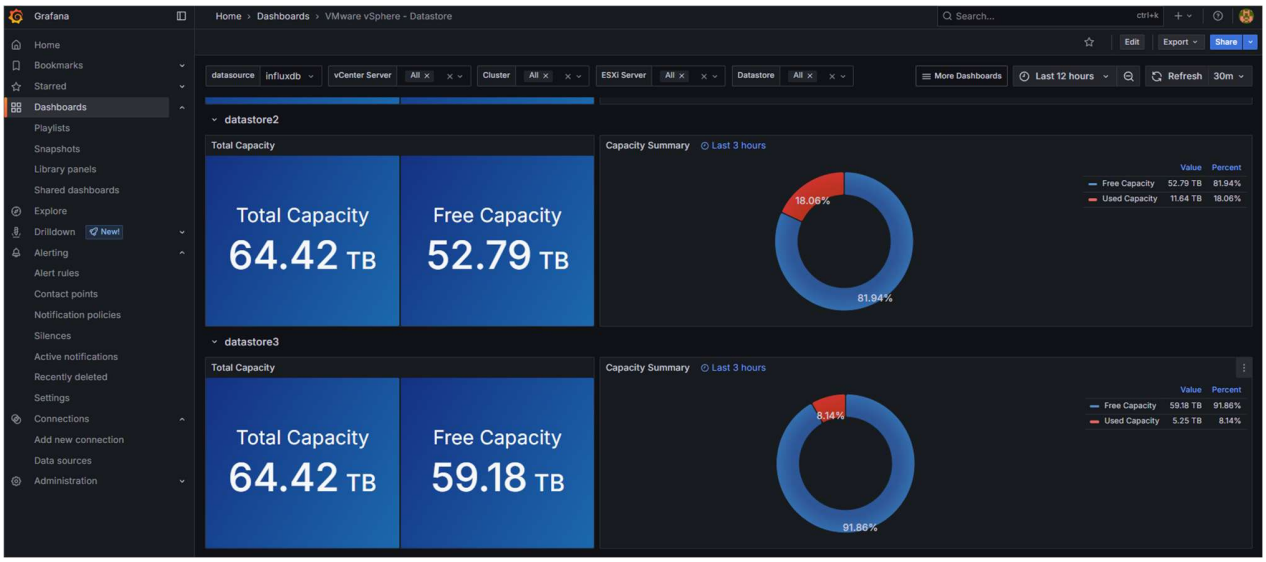


Рисунок 3.22 – Моніторинг високонавантажених мережевих сховищ великої місткості

На рис. 3.22 продемонстровано роботу системи з основними масивами даних інфраструктури (`datastore2` та `datastore3`), [26]. Аналіз цих панелей дозволяє зробити наступні висновки:

Масштабованість моніторингу: система коректно обробляє та візуалізує дані великих об'ємів (понад 64 TB на кожне сховище), що підтверджує стабільність роботи стеку InfluxDB+Grafana з високою щільністю метрик.

Аналіз заповненості: * Для `datastore2` зафіксовано використання 11.64 TB (18.06%), що свідчить про активне завантаження дискового простору робочими даними.

Для `datastore3` показник використання складає 5.25 TB (8.14%), що вказує на наявність значного резерву для подальшого розгортання сервісів.

Оперативне планування: чітка візуалізація співвідношення вільного та зайнятого місця (Free Capacity 52.79 TB та 59.18 TB відповідно) надає адміністратору можливість точно прогнозувати терміни розширення апаратної частини ЦОД.

Used Capacity Trend: індикатор використання у 1.47% (1.48 GB) підтверджує мінімальне поточне навантаження на обраний локальний диск, що є оптимальним для стабільної роботи системних логів та файлів конфігурації.

3.9 Налаштування системи автоматичного оповіщення про критичні стани

Для забезпечення безперервного контролю над інфраструктурою та мінімізації часу реакції на інциденти, у розробленій системі було реалізовано підсистему алертингу (Alerting), [27]. Це дозволяє автоматизувати процес виявлення аномалій і миттєво сповіщати адміністратора через зовнішні канали зв'язку. Загальну логіку функціонування підсистеми оповіщень у Grafana представлено на рис. 3.23.

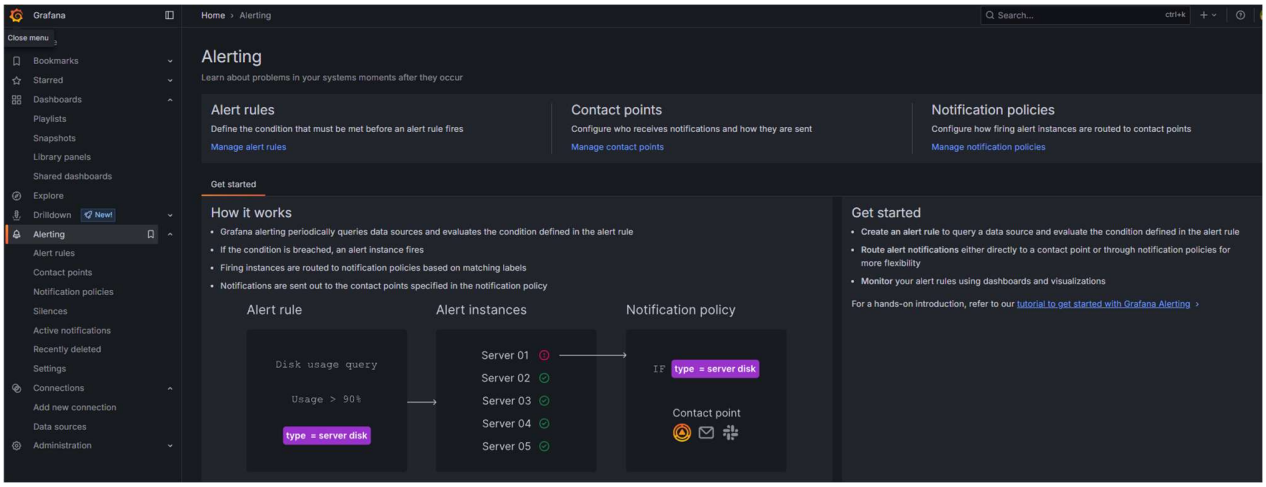


Рисунок 3.23 – Принцип роботи системи сповіщень Grafana Alerting

Як продемонстровано на рис. 3.23, процес моніторингу складається з трьох ключових етапів: перевірка правил (Alert rule), виявлення інстансів, що порушують межі (Alert instances), та спрямування сповіщення [6] згідно з політикою (Notification policy).

3.9.1 Конфігурування каналів зв'язку та контактних точок

Першим практичним кроком стала реєстрація контактних точок (Contact Points). Основним каналом було обрано месенджер Telegram. На рис. 3.23 наведено перелік сконфігурованих точок доступу, а на рис. 3.24 – технічні параметри інтеграції.

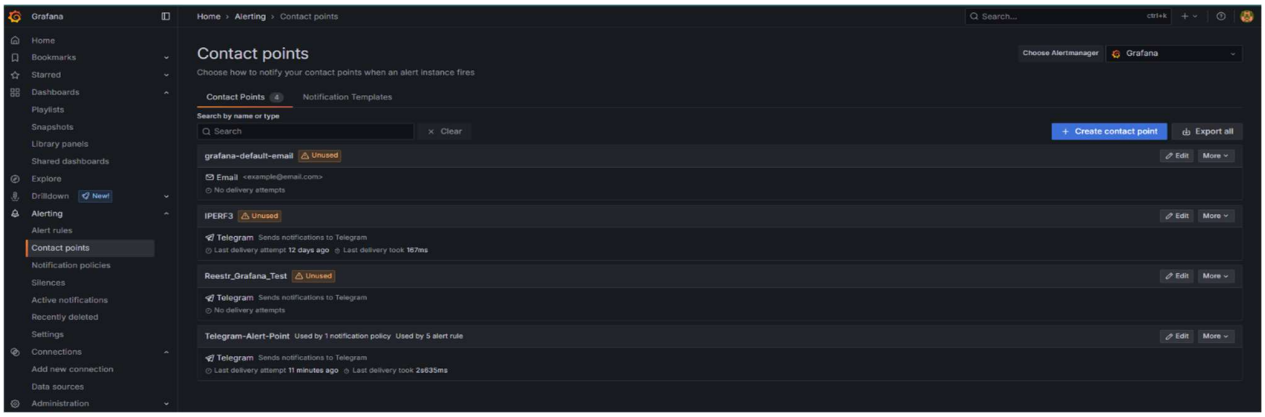


Рисунок 3.24 – Перелік налаштованих каналів сповіщення (Telegram-Alert-Point)

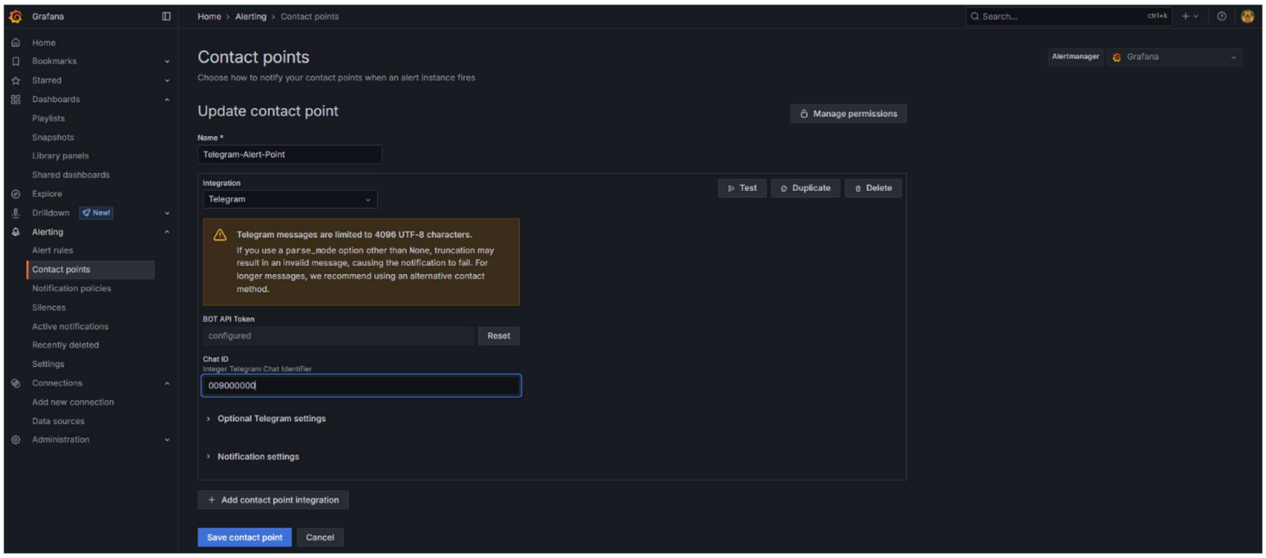


Рисунок 3.24 – Параметри інтеграції з Telegram Bot API

3.9.2 Формування правил моніторингу та логіка обробки інцидентів

Було впроваджено контроль за такими параметрами: навантаження на CPU та RAM віртуальних навантаження на CPU та RAM віртуальних машин і хостів, доступність вузлів та стан дискових сховищ, [28]. Загальний перелік активних правил представлено на рис. 3.26.

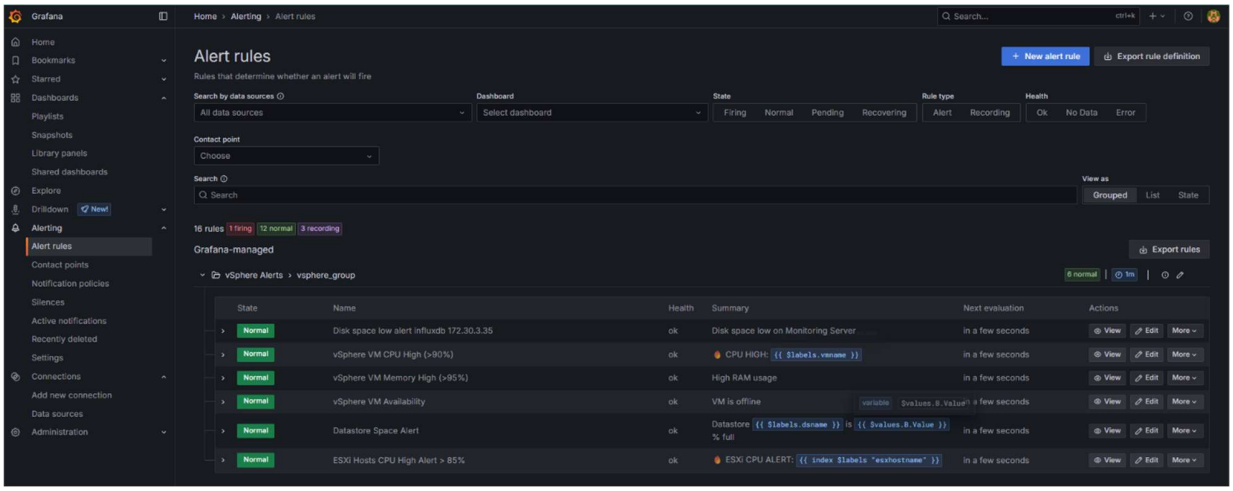


Рисунок 3.26 – Перелік налаштованих правил для контролю стану інфраструктури

Для вибору даних використовується мова запитів Flux (рис. 3.27), що дозволяє динамічно відстежувати кожен вузол окремо, [29]. Повний текст складного запиту наведено у додатку Г.

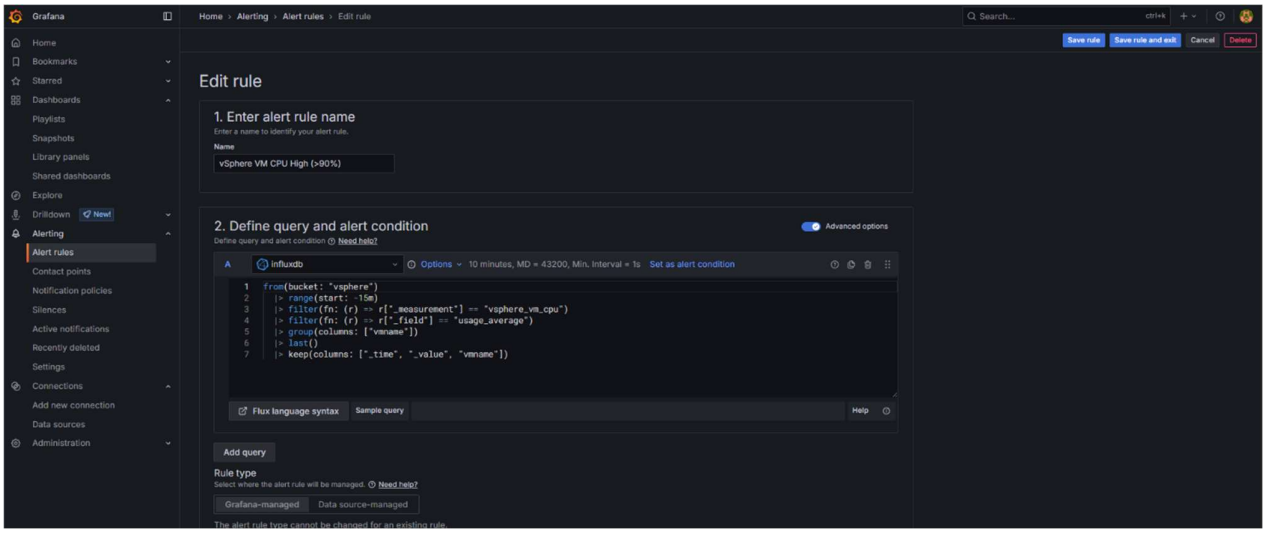


Рисунок 3.27 – Використання Flux-запиту для агрегації даних про навантаження

Для забезпечення високої точності роботи системи алертингу було проведено детальне налаштування параметрів перевірки метрик.

На рис. 3.26 показано інтерфейс обробки телеметрії. Для зменшення навантаження на систему візуалізації та згладжування пікових значень завантаження CPU використовується функція aggregateWindow, яка обчислює середнє значення за 5-хвилинні інтервали. Приклад реалізації такої фільтрації та агрегації для конкретного вузла esxi-node-01 наведено у Додатку Д.

На рис. 3.28 представлено інтерфейс налаштування інтервалу оцінки стану (Evaluation behavior).

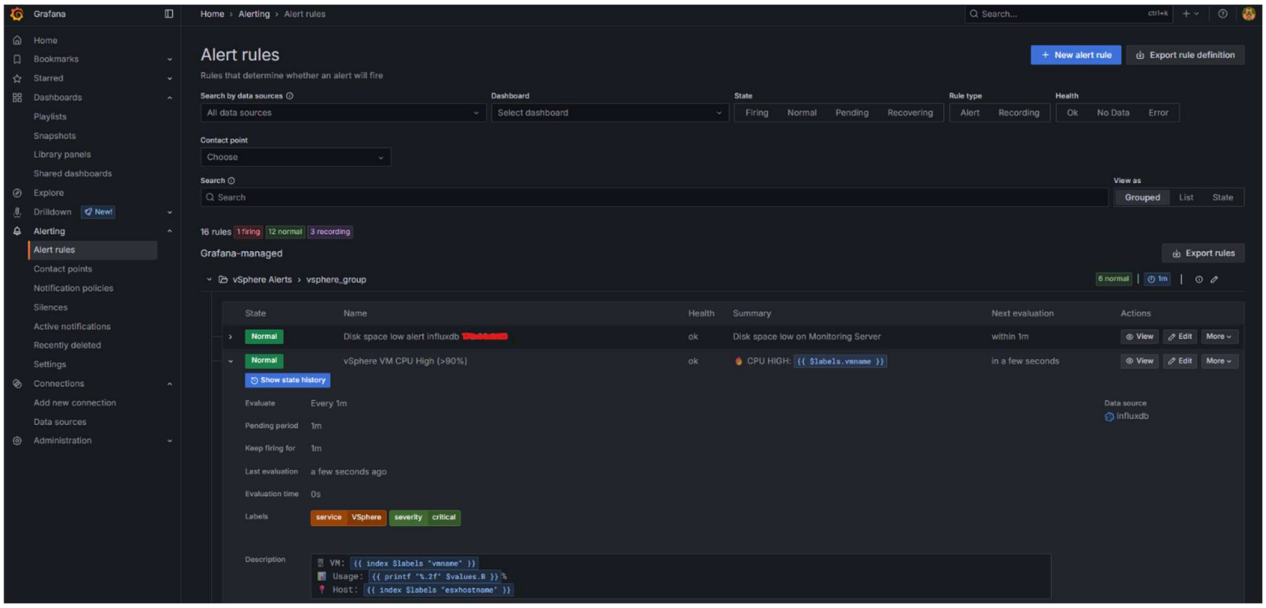


Рисунок 3.28 – Параметри часових інтервалів перевірки правил

Згідно з рис. 3.28, параметр «For» встановлено на 1 хвилину. Це дозволяє системі ігнорувати короткочасні сплески навантаження, які не впливають на загальну стабільність вузла. Далі на рис. 3.29 визначено логіку поведінки системи при виникненні технічних помилок або відсутності даних у базі.

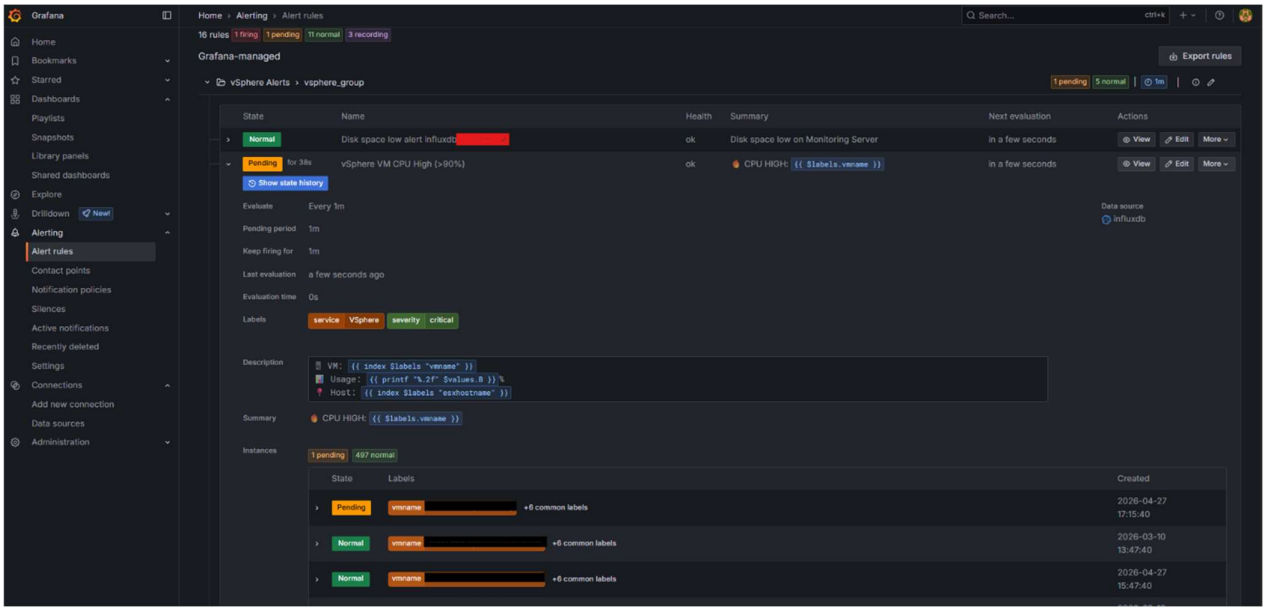


Рисунок 3.29 – Конфігурування обробки помилок та відсутності даних

Наступним кроком стало додавання метаданих до кожного правила. На рисунках 3.29 та 3.30 продемонстровано процес призначення міток (Labels) та

анотацій, що дозволяє автоматично додавати технічну інформацію до тексту сповіщення. [30]

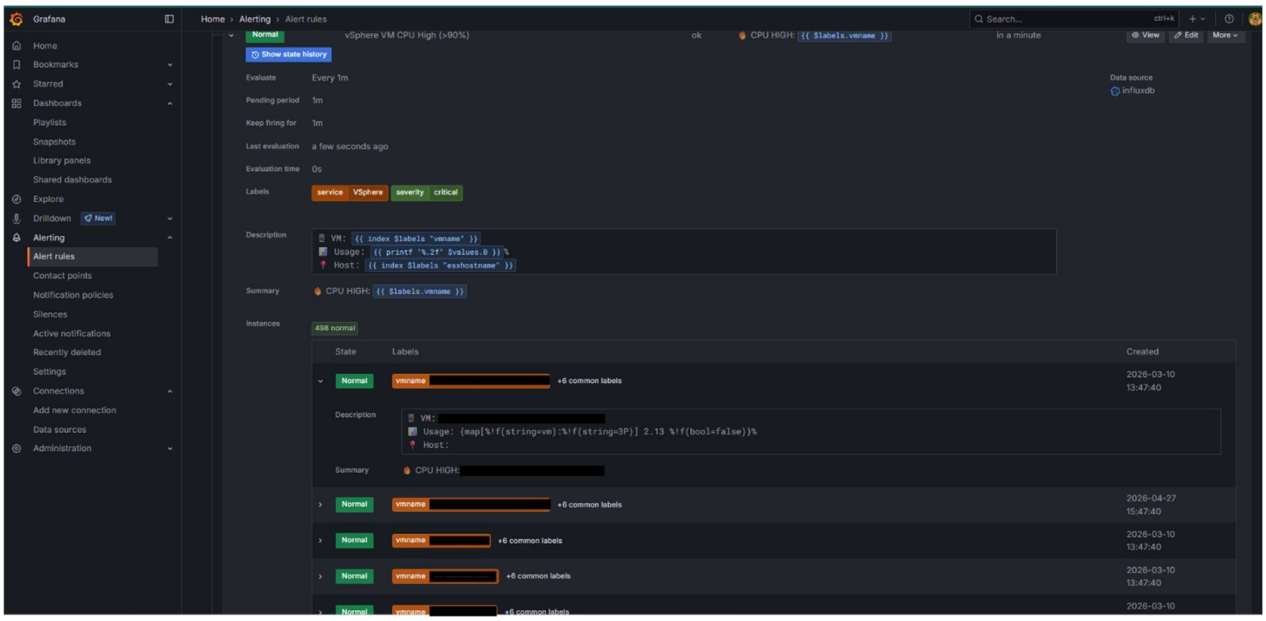


Рисунок 3.30 – Призначення додаткових міток для ідентифікації вузлів

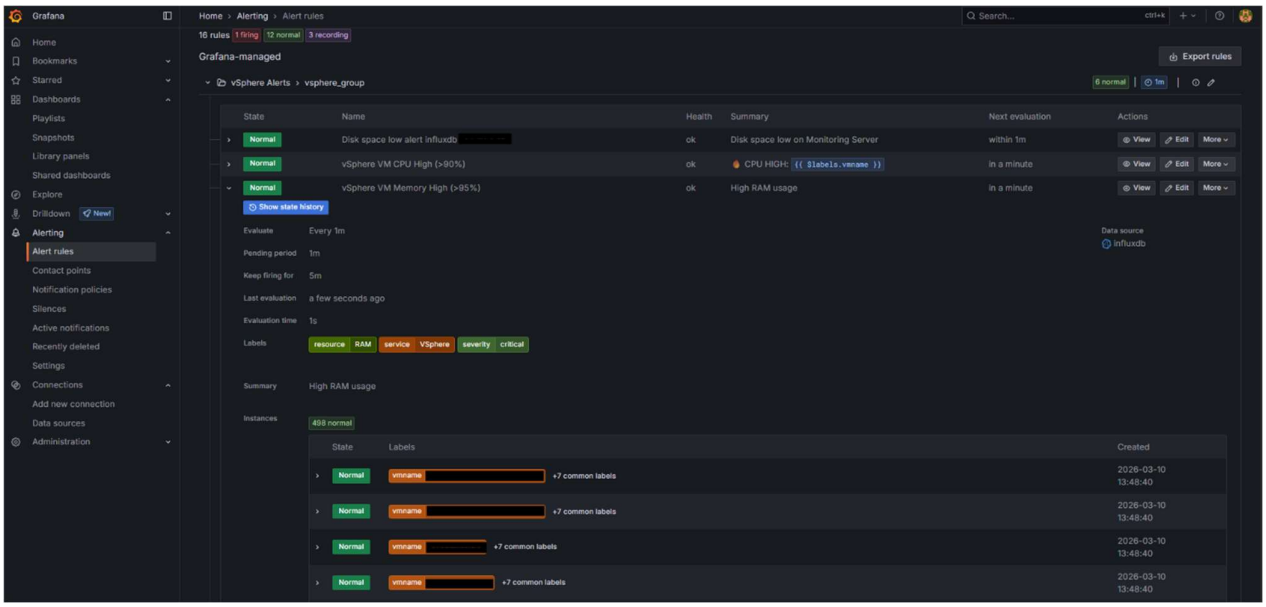


Рисунок 3.31 – Формування тексту повідомлення за допомогою шаблонів анотацій

Такий підхід (рис. 3.31) забезпечує гнучкість: система самостійно підставляє ім'я вузла, що дозволяє адміністратору миттєво зрозуміти локалізацію проблеми без переходу в консоль Grafana.

3.9.3 Політики маршрутизації та оптимізація сповіщень

Останнім етапом конфігурування підсистеми алертингу стало налаштування політик сповіщень (Notification Policies). Це дозволяє системі визначати, до якої саме контактної точки відправляти конкретний тип повідомлення. На рис. 3.32 представлено ієрархію розсилки.

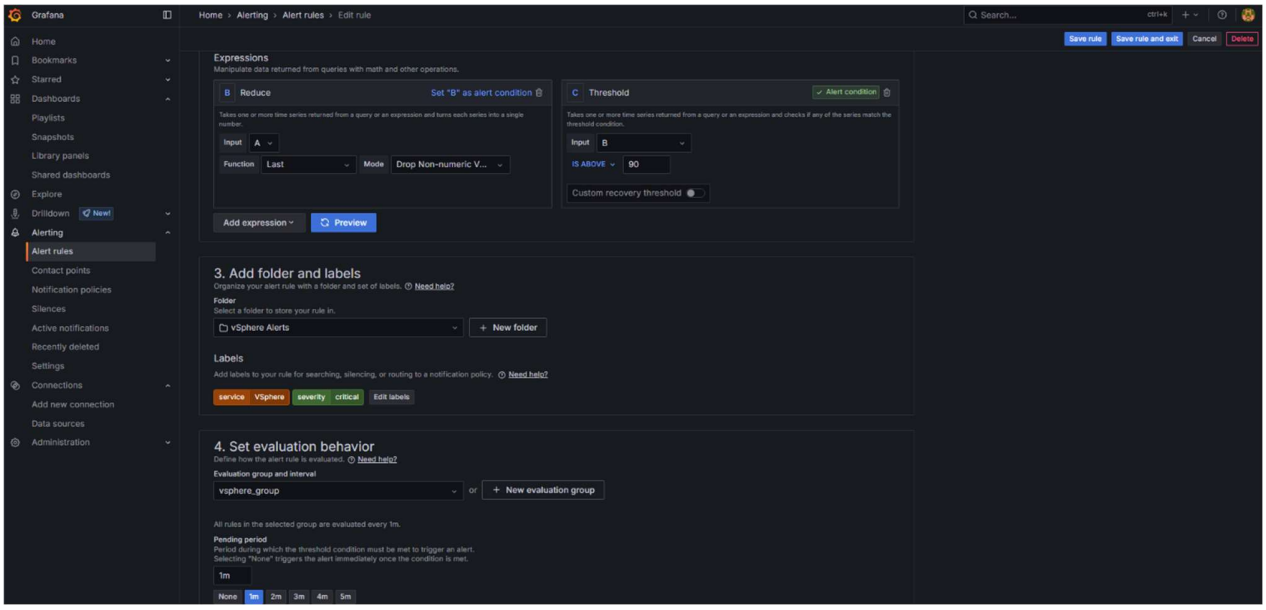


Рисунок 3.32 – Налаштування політик маршрутизації сповіщень

Як показано на рис. 3.32, для всіх правил групи vSphere Alerts визначено канал Telegram-Alert-Point. Важливим аспектом є можливість відстеження історії спрацювань безпосередньо в інтерфейсі Grafana [31], (рис. 3.33 та 3.34).

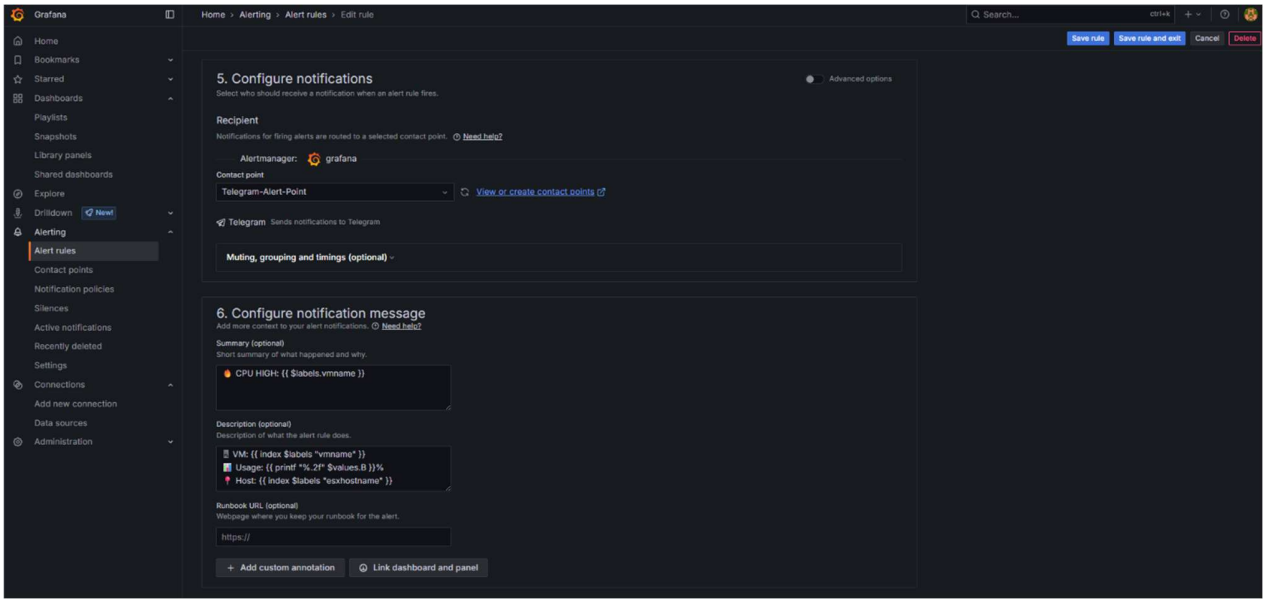


Рисунок 3.33 – Журнал станів алерту: історія переходів між Normal, Pending та Firing

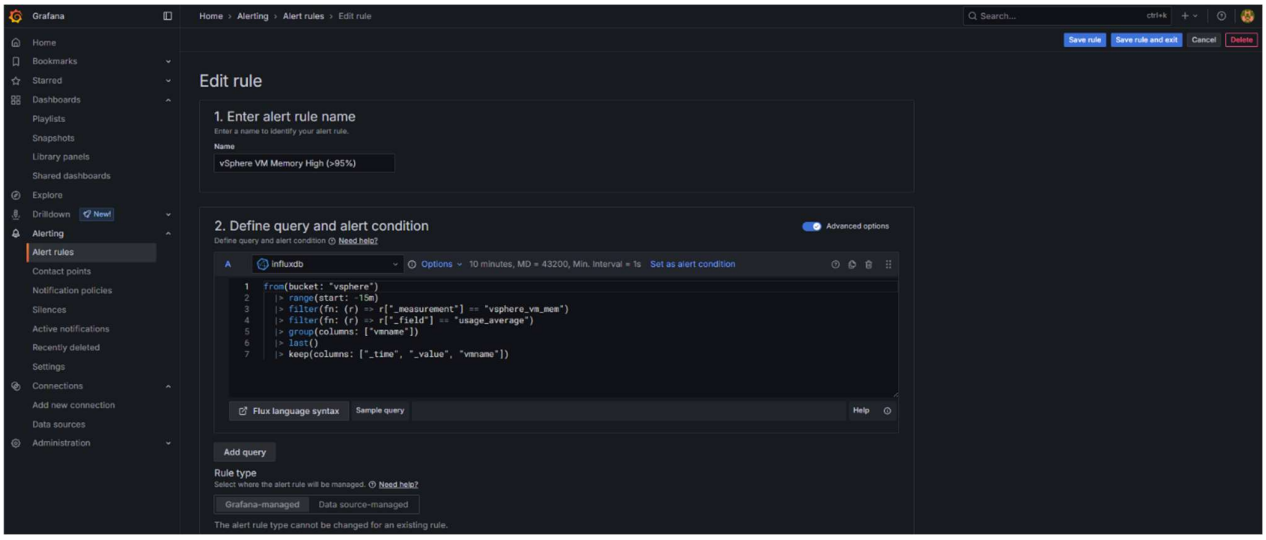


Рисунок 3.34 – Графічна інтерпретація моменту порушення порогу та тривалості інциденту

Для запобігання ефекту «втоми від сповіщень» було налаштовано часові інтервали групування. На рис. 3.35 представлено структуру кореневої політики, де визначено час очікування для збору кількох подій в одне повідомлення.

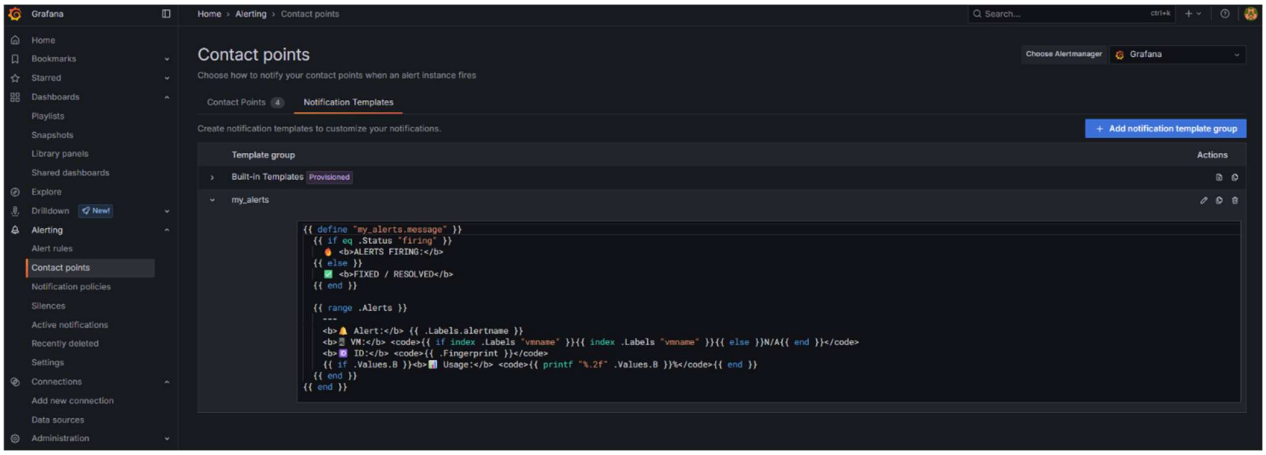


Рисунок 3.35 – Конфігурування часових інтервалів Group wait та Repeat interval

Для виключення хибних сповіщень під час планового обслуговування вузлів було використано механізм Mute Timings. На рис. 3.36 та 3.37 продемонстровано створення розкладів «тиші».

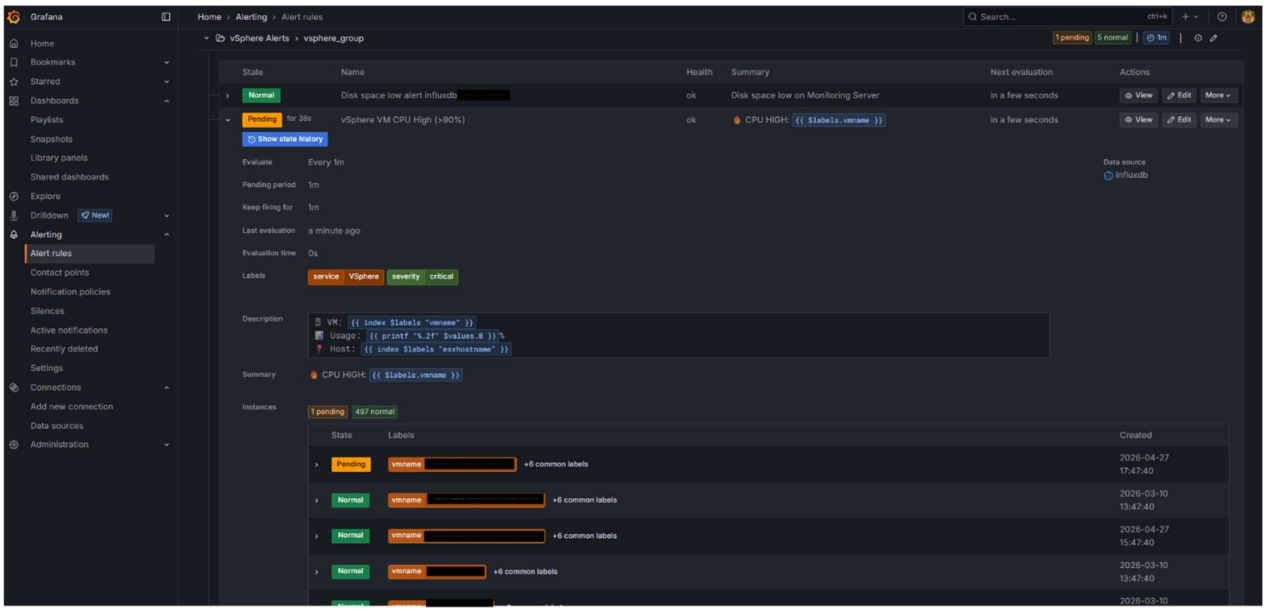


Рисунок 3.36 – Створення розкладу Mute Timings для регламентних робіт

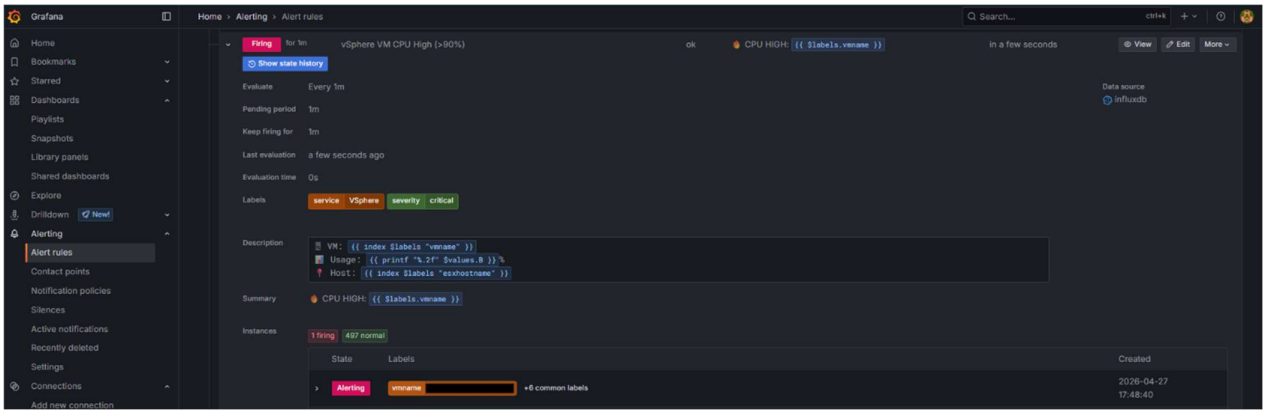


Рисунок 3.37 – Деталізація часових проміжків активності режиму тиші

На завершення налаштування було проведено перевірку маршрутизації (рис. 3.38 та 3.39), що гарантує доставку сповіщень саме до Telegram-бота.

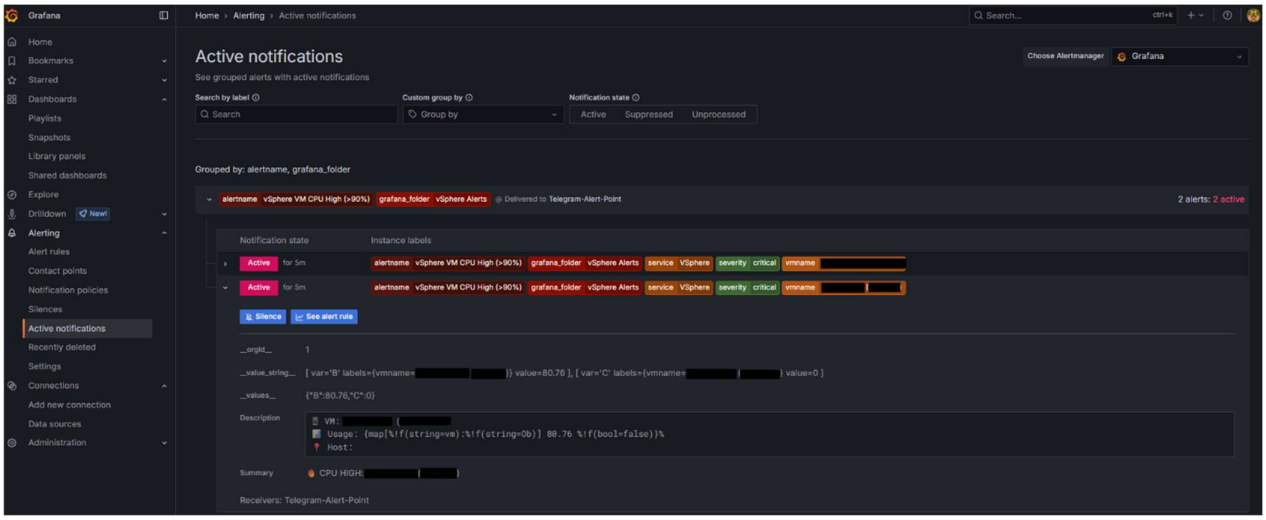


Рисунок 3.38 Налаштування маршрутів для сповіщенньь

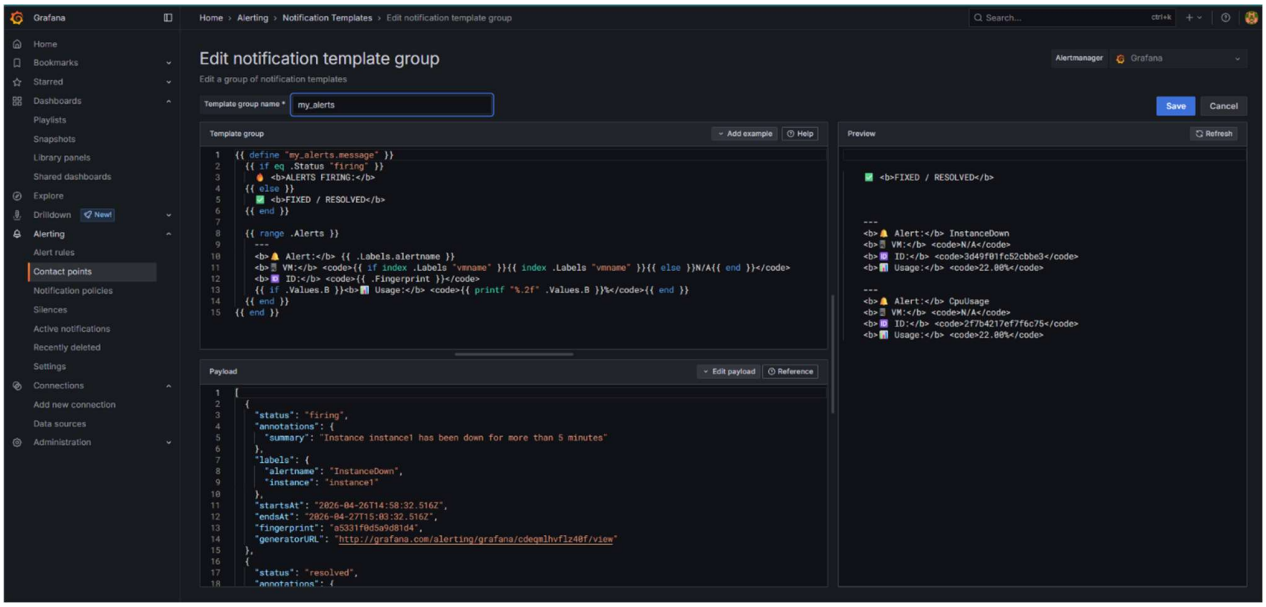


Рисунок 3.39 – Верифікація маршрутів та вибір специфічних міток для сповіщень

3.9.4 Аналіз функціонування системи в робочих умовах

Працездатність усього налаштованого стеку було перевірено шляхом моніторингу інфраструктури в періоди реальної активності. На рис. 3.40 представлено тестове повідомлення, а на наступних рисунках – реальні цикли інцидентів.

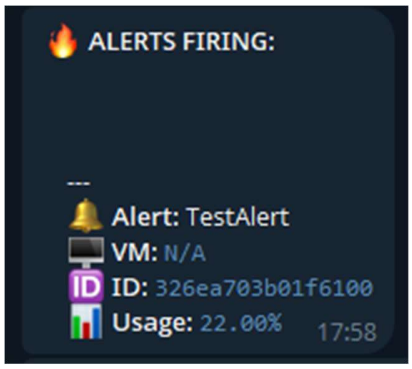


Рисунок 3.40 – Тестове підтвердження зв'язку з Telegram API

Для демонстрації повного циклу обробки інциденту на рис. 3.41 наведено сповіщення про дефіцит оперативної пам'яті на одному з вузлів.

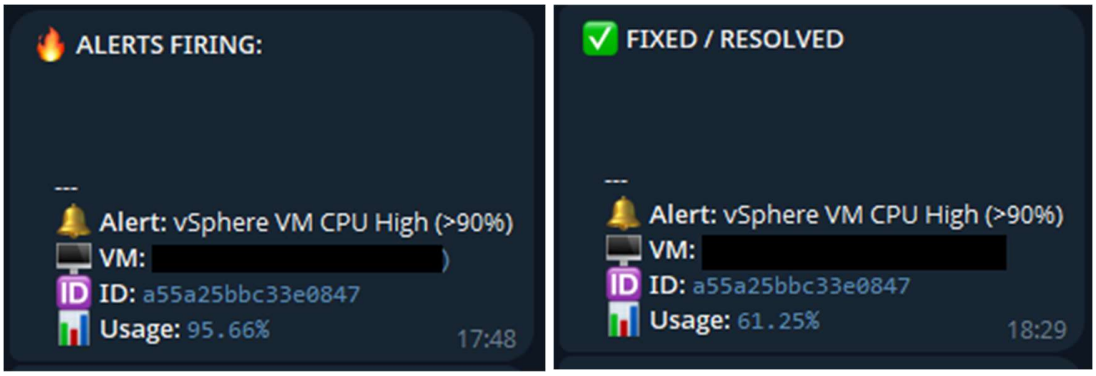


Рисунок 3.41 – Життєвий цикл алерту RAM: виявлення проблеми (ліворуч) та її вирішення (праворуч)

Аналогічно на рис. 3.42 зафіксовано пікове навантаження на процесор вузла та подальше повернення метрики до нормальних значень.

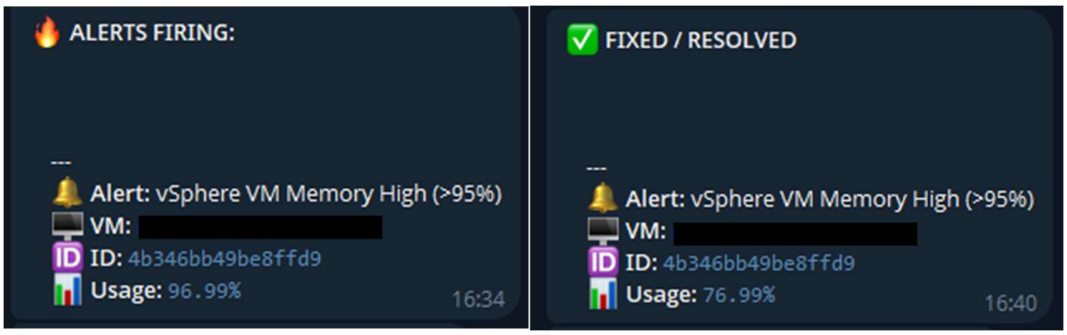


Рисунок 3.42 – Життєвий цикл алерту CPU: статус FIRING та автоматичний статус RESOLVED

Загальний вигляд журналу сповіщень у месенджері (рис. 3.43) дозволяє адміністратору оперативно аналізувати хронологію подій та стабільність інфраструктури без доступу до стаціонарного робочого місця.

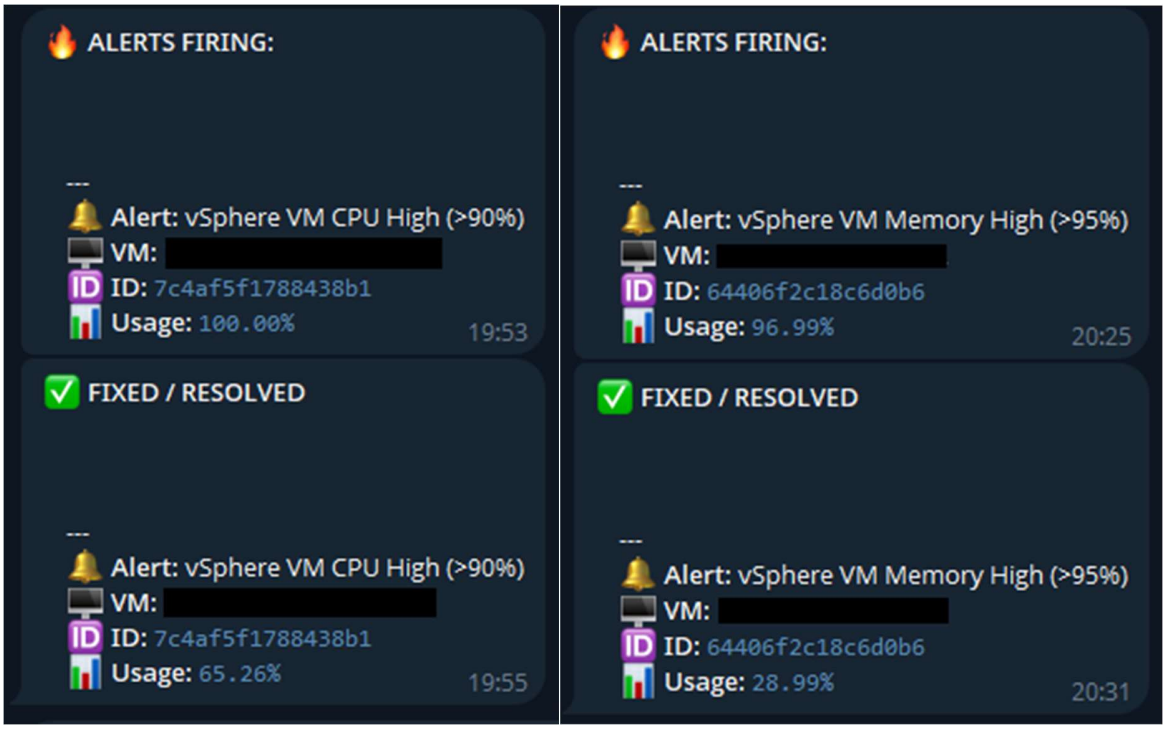


Рисунок 3.43 – Підсумковий вигляд робочого каналу сповіщень у Telegram

ВИСНОВКИ ДО РОЗДІЛУ 3

Реалізована система моніторингу на основі Telegraf, InfluxDB та Grafana повністю вирішує задачу прозорого та безперервного контролю віртуалізованої інфраструктури. [33]. Завдяки використанню набору спеціалізованих дашбордів було досягнуто:

Повної видимості: від загального стану кластерів до конкретних показників затримок на окремих дисках.

Аналітичної глибини: можливість відстеження критичних параметрів, таких як CPU Ready Time та IOPS Latency.

Ефективності управління: скорочення часу на пошук несправностей та планування ресурсів.

Мобільності: впровадження системи сповіщень через Telegram дозволило адміністратору отримувати дані про стан ЦОД у режимі реального часу.

4. АНАЛІЗ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Оцінка технічної стабільності та продуктивності

У ході проведення випробувань розгорнутої системи було проаналізовано її вплив на ресурси інфраструктури. Завдяки використанню контейнеризації Docker та оптимізованого агента Telegraf, споживання ресурсів хоста, на якому функціонує моніторинговий стек, залишається мінімальним:

Використання CPU не перевищує 2–3% від загальної потужності.

Споживання оперативної пам’яті стеком TIG стабілізувалося на позначці до 1.5 ГБ, що є допустимим показником для систем такого класу.

Використання бази даних InfluxDB дозволило ефективно зберігати часові ряди завдяки алгоритмам стиснення, що мінімізує витрати дискового простору, [36].

4.2 Аналіз операційної ефективності та швидкості реагування

Ключовим показником ефективності впровадженої системи є скорочення часу виявлення інцидентів (Mean Time To Detect – MTTD). У таблиці 4.1 наведено порівняльний аналіз стандартного підходу та розробленої системи [3, 6, 7].

Таблиця 4.1 – Порівняння швидкості реакції на інциденти

Параметр порівняння	**Стандартний метод (vSphere Client)**	**Розроблена система (TIG + Telegram)**
Спосіб отримання даних	Ручна перевірка консолі	Автоматичне сповіщення
Середній час виявлення	15–60 хв (залежно від чергування)	30–60 секунд
Доступність даних	Лише з робочого місця (VPN/LAN)	Будь-яка точка (мобільний пристрій)
Візуалізація	Таблична, обмежена	Інтерактивні графіки (Grafana)

Впровадження Telegram-бота дозволило адміністратору отримувати критичні дані миттєво, що фактично нівелює ризик тривалого простою сервісів через непомічену вчасно проблему.

4.3 Економічний аспект та переваги відкритого ПЗ

Вибір стеку Telegraf, InfluxDB та Grafana забезпечує значну економічну перевагу, оскільки всі компоненти поширюються за ліцензіями Open Source. Це дозволяє:

Уникнути витрат на ліцензування спеціалізованих пропрієтарних рішень (наприклад, VMware vRealize Operations), вартість яких може становити тисячі доларів на рік.

Гнучкість налаштувань: відсутність прив’язки до одного вендора дозволяє за потреби розширювати систему моніторингу на інші типи обладнання чи мережеві пристрої.

Зниження вимог до персоналу: завдяки зручним дашбордам та зрозумілим сповіщенням у месенджері, первинний аналіз стану системи може виконувати навіть черговий персонал без глибоких знань vSphere API.

ВИСНОВКИ ДО РОЗДІЛУ 4

Проведений аналіз підтвердив, що розроблена система відповідає всім сучасним вимогам до засобів моніторингу ЦОД. Вона забезпечує високу швидкість реакції на відмови, має низькі системні вимоги та є економічно вигідною завдяки використанню вільного програмного забезпечення.

ВИСНОВКИ

У бакалаврській дипломній роботі проведено комплексне дослідження та успішно вирішено науково-практичне завдання щодо створення автоматизованої системи моніторингу віртуалізованої інфраструктури. За результатами роботи сформульовано наступні висновки:

Аналіз сучасного стану систем моніторингу підтвердив, що для великих масивів даних, які генерує VMware vSphere, класичні реляційні бази даних є малоефективними. Обґрунтовано доцільність переходу до архітектур на основі часових рядів (Time Series), що дозволяє обробляти тисячі метрик щосекунди без затримок у візуалізації.

Науково обґрунтовано вибір програмного стеку TIG. Використання агента Telegraf дозволило мінімізувати навантаження на гіпервізори при зборі даних. База даних InfluxDB забезпечила ефективне стиснення інформації, а мова запитів Flux дозволила реалізувати складну математичну обробку метрик безпосередньо на етапі вибірки, що підвищило продуктивність системи в цілому.

Застосування технології контейнеризації Docker дозволило уніфікувати процес розгортання компонентів системи. Це забезпечило ізоляцію сервісів, спростило процедуру резервного копіювання конфігурацій та дозволило швидко масштабувати систему моніторингу при додаванні нових вузлів до інфраструктури ЦОД.

Створено багаторівневу систему візуалізації. Налаштовані інтерактивні дашборди в Grafana відображають як загальний стан кластерів, так і детальні показники окремих об'єктів. Зокрема, реалізовано контроль дискових сховищ великого об'єму, що дозволяє прогнозувати час заповнення ємностей та уникати зупинки критичних сервісів.

Розроблено та впроваджено інтелектуальну систему алертингу. Налаштування політик сповіщень та режимів тиші (Mute Timings) дозволило

уникнути надмірної кількості повідомлень та зосередити увагу адміністратора лише на критичних інцидентах. Використання Telegram-бота як кінцевої точки доставки забезпечило мобільність управління: час від моменту виникнення перевантаження CPU чи RAM до отримання сповіщення не перевищує 30-60 секунд.

Результати тестування системи в робочих умовах (зокрема під час вечірніх пікових навантажень) підтвердили її високу надійність та точність. Система успішно ідентифікувала вузли, що потребували оптимізації ресурсів, що дозволило підвищити стабільність усієї віртуальної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гілл Ч. Хмарні обчислення: принципи та архітектура. – К.: Техніка, 2022. – 320 с. *(Для теоретичного розділу про віртуалізацію)*.
2. Docker documentation. Orientation and setup. URL: https://docs.docker.com/get-started/
3. Telegraf Documentation. vSphere Input Plugin. URL: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/vsphere
4. InfluxData Documentation. InfluxDB OSS 2.x Documentation. URL: https://docs.influxdata.com/influxdb/v2/
5. Flux Language Reference. Get started with Flux. URL: https://docs.influxdata.com/flux/v0.x/
6. Grafana Labs. Grafana Alerting documentation. URL: https://grafana.com/docs/grafana/latest/alerting/
7. VMware vSphere Documentation. Monitoring Performance and Health. URL: https://docs.vmware.com/en/VMware-vSphere/index.html
8. Telegram Bot API. Official documentation for developers. URL: https://core.telegram.org/bots/api
9. Chacon S., Straub B. Pro Git. – Apress, 2014. – 456 p. *(Якщо згадував версіонування конфігів)*.
10. Postman Learning Center. Introduction to APIs. URL: https://learning.postman.com/docs/publishing-your-api/api-documentation-overview/
11. InfluxDB OSS 2.7 Documentation. URL: https://docs.influxdata.com/influxdb/v2.7/
12. Telegraf vSphere Input Plugin Specification. InfluxData GitHub Repository. URL: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/vsphere

13. Time Series Data and InfluxDB: Under the hood. InfluxData White Papers. URL: https://www.influxdata.com/time-series-database/

14. VMware vSphere Documentation. Troubleshooting Resource Contention. VMware by Broadcom. URL: https://docs.vmware.com/en/VMware-vSphere/8.0/vsphere-monitoring-performance/GUID-62423EDF-B552-4752-9A4B-BAE932A510A2.html

15. VMware vSphere Monitoring and Performance Guide. Understanding Disk Latency Metrics. VMware Inc. URL: https://docs.vmware.com/en/VMware-vSphere/8.0/vsphere-monitoring-performance/GUID-37651030-9759-462B-9B2F-93105B036746.html

16. Posta C. Cloud-Native Infrastructure: Patterns for Scalable Infrastructure and Applications in a Dynamic Environment. O'Reilly Media, 2023. 240 p.

17. Docker Compose Overview and Best Practices. Docker Documentation. URL: https://docs.docker.com/compose/intro/overview/

18. Turnbull J. The Art of Monitoring. Turnbull Press, 2016. 434 p.

19. Marsh S. Docker Compose in Action. Systematic deployment of multi-container applications. Manning Publications, 2023. 280 p.

20. Bentley J. Docker Cloud-Native Monitoring. Cloud-Native Operations with Infrastructure as Code. Manning Publications, 2022. 340 p.

21. Grafana Labs. Grafana Documentation: Fundamentals of Data Visualization and Dashboarding. Grafana Labs, 2024. URL: https://grafana.com/docs/

22. Grafana Labs. VMware vSphere - Datastore Dashboard by gallovad (ID: 8159). Grafana Dashboards, 2018. URL: https://grafana.com/grafana/dashboards/8159

23. Grafana Labs. VMware vSphere - Hosts: Dashboard for ESXi host monitoring (ID: 8154). Grafana Dashboards, 2018. URL: https://grafana.com/grafana/dashboards/8154

24. Grafana Labs. VMware vSphere - Overview: Infrastructure health and resource summary (ID: 8162). Grafana Dashboards, 2018. URL: https://grafana.com/grafana/dashboards/8162

25. Grafana Labs. VMware vSphere - VMs: Virtual machine performance tracking (ID: 8157). Grafana Dashboards, 2018. URL: https://grafana.com/grafana/dashboards/8157

26. Grafana Labs. Visualizing Storage Metrics: Pie Charts and Stat Panels in Grafana. Grafana Documentation, 2024. URL: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/

27. Grafana Labs. Alerting Rules and Notifications: Integrating with External Services. Grafana Documentation, 2024. URL: https://grafana.com/docs/grafana/latest/alerting/

28. Marshall N., et al. Mastering VMware vSphere 7: Optimize Infrastructure, Monitor Performance, and Manage Virtual Machines. Wiley, 2021. 832 p.

29. Naeem S. Data Analysis with InfluxDB: Querying and processing time-series data with Flux. Packt Publishing, 2023. 240 p.

30. Joyner J. Intelligent Alerting: A Guide to Reducing Alert Fatigue with Labels and Annotations. InfluxData Technical Blog, 2023. URL: https://www.influxdata.com/blog/

31. Grafana Labs. Managing Alert State History and Notification Policies. Grafana Documentation, 2024. URL: https://grafana.com/docs/grafana/latest/alerting/fundamentals/state-history/

32. Brenninkmeijer J. Monitoring with Grafana: Building Hierarchical Dashboards for Enterprise Infrastructure. O'Reilly Media, 2022. 156 p.

33. Chitnis A., et al. Hands-On Infrastructure Monitoring with Prometheus and Grafana: Implement custom and industry-standard monitoring solutions. Packt Publishing, 2021. 354 p. (Note: Chapter on Telegraf and InfluxDB integration).

34. Docker Documentation. Docker Compose overview and multi-container orchestration. Docker Docs, 2024. URL: https://docs.docker.com/compose/ (дата звернення: 07.05.2026).

35. VMware. vSphere Automation API Reference Guide. VMware Docs, 2023. URL: https://developer.broadcom.com/xapis/vsphere-automation-api/

36. InfluxData. InfluxDB Storage Engine and TSM File Structure: Compression and Compaction Algorithms. InfluxData Documentation, 2024. URL: https://docs.influxdata.com/influxdb/v2/reference/internals/storage-engine/

37. Jain R. The Art of Computer Systems Performance Analysis: Techniques for
Experimental Design, Measurement, Simulation, and Modeling. Wiley, 1991. 720 p.

38. Walpole R. E., et al. Probability & Statistics for Engineers & Scientists. Pearson, 2016. 848 p.

39. Pruim R. Foundations and Applications of Statistics: An Introduction using R. American Mathematical Society, 2018. 823 p. (Section on Standard Deviation and Outlier Detection).

40. Montgomery D. C., et al. Introduction to Linear Regression Analysis. Wiley, 2021. 720 p.

Додаток А

Лістинг програмного коду на мові JavaScript для обробки сповіщень.

```
JavaScript
capacity = from(bucket: "vsphere")
  |> range(start: v.timeRangeStart, stop:
v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] ==
"vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "capacity_latest")
free = from(bucket: "vsphere")
  |> range(start: v.timeRangeStart, stop:
v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] ==
"vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "freespace_latest")
join(tables: {cap: capacity, fr: free}, on: ["_time",
"datastore", "esxhost"])
  |> map(fn: (r) => ({
      _time: r._time,
      _value: float(v: r._value_fr) / float(v:
r._value_cap) * 100.0,
      _field: "free_percent"
    }))
```

Додаток Б

Конфігураційний файл агента збору метрик telegraf.conf.

```
# Configuration for telegraf agent

[agent]

  ## Default data collection interval for all inputs

  interval = "10s"

  ## Rounds collection interval to 'interval'

  ## ie, if interval="10s" then always collect on :00, :10, :20,
etc.

  round_interval = true


  ## Telegraf will send metrics to outputs in batches of at most

  ## metric_batch_size metrics.

  ## This controls the size of writes that Telegraf sends to output
plugins.

  metric_batch_size = 1000


  ## For failed writes, telegraf will cache metric_buffer_limit
metrics for each

  ## output, and will flush this buffer on a successful write.
Oldest metrics

  ## are dropped first when this buffer fills.

  ## This buffer only fills when writes fail to output plugin(s).

  metric_buffer_limit = 10000

  ## Collection jitter is used to jitter the collection by a random
amount.

  ## Each plugin will sleep for a random time within jitter before
collecting.

  ## This can be used to avoid many plugins querying things like
sysfs at the

  ## same time, which can have a measurable effect on the system.

  collection_jitter = "0s"
```

```
  ## Default flushing interval for all outputs. Maximum
flush_interval will be

  ## flush_interval + flush_jitter

  flush_interval = "10s"

  ## Jitter the flush interval by a random amount. This is
primarily to avoid

  ## large write spikes for users running a large number of
telegraf instances.

  ## ie, a jitter of 5s and interval 10s means flushes will happen
every 10-15s

  flush_jitter = "0s"

  ## By default or when set to "0s", precision will be set to the
same

  ## timestamp order as the collection interval, with the maximum
being 1s.

  ##   ie, when interval = "10s", precision will be "1s"

  ##       when interval = "250ms", precision will be "1ms"

  ## Precision will NOT be used for service inputs. It is up to
each individual

  ## service input to set the timestamp at the appropriate
precision.

  ## Valid time units are "ns", "us" (or "µs"), "ms", "s".

  precision = ""

  ## Logging configuration:

  ## Run telegraf with debug log messages.

  debug = false

  ## Run telegraf in quiet mode (error log messages only).

  quiet = false

  ## Specify the log file name. The empty string means to log to
stderr.

  logfile = ""
```

```
 ## Override default hostname, if empty use os.Hostname()
  hostname = ""
  ## If set to true, do no set the "host" tag in the telegraf
agent.
  omit_hostname = false
[[outputs.influxdb_v2]]
  ## The URLs of the InfluxDB cluster nodes.
  ##
  ## Multiple URLs can be specified for a single cluster, only ONE
of the
  ## urls will be written to each interval.
  ## urls exp: http://127.0.0.1:8086
  urls = ["http://influxdb:8086"]

  ## Token for authentication.
  token = 'nWW3swgUHbp3yqxorX7htz……..CvaqQQ=='

  ## Organization is the name of the organization you wish to write
to; must exist.
  organization = "telegraf"

  ## Destination bucket to write into.
  bucket = "esxi"

  insecure_skip_verify = true

[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
```

```
## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU
states.
  report_active = false
[[inputs.disk]]
  ## By default stats will be gathered for all mount points.
  ## Set mount_points will restrict the stats to only the specified
mount points.
  # mount_points = ["/"]
  ## Ignore mount points by filesystem type.
  ignore_fs = ["tmpfs", "devtmpfs", "devfs", "overlay", "aufs",
"squashfs"]
[[inputs.diskio]]
[[inputs.mem]]
[[inputs.net]]
[[inputs.processes]]
[[inputs.swap]]
[[inputs.system]]
# Read metrics from one or many vCenters
[[inputs.vsphere]]
  ## List of vCenter URLs to be monitored. These three lines must
be uncommented
  ## and edited for the plugin to work.
  vcenters = [ "https://172.10.10.10/sdk" ]
  username = " username "
  password = " password "
```

```
insecure_skip_verify = true  # Це змусить Telegraf ігнорувати помилку сертифіката

  ## VMs

  ## Typical VM metrics (if omitted or empty, all metrics are collected)

  # vm_include = [ "/*/vm/**"] # Inventory path to VMs to collect (by default all are collected)

  # vm_exclude = [] # Inventory paths to exclude

  vm_metric_include = [

    "cpu.demand.average",

    "cpu.idle.summation",

    "cpu.latency.average",

    "cpu.readiness.average",

    "cpu.ready.summation",

    "cpu.run.summation",

    "cpu.usagemhz.average",

    "cpu.used.summation",

    "cpu.wait.summation",

    "mem.active.average",

    "mem.granted.average",

    "mem.latency.average",

    "mem.swapin.average",

    "mem.swapinRate.average",

    "mem.swapout.average",

    "mem.swapoutRate.average",

    "mem.usage.average",

    "mem.vmmemctl.average",

    "net.bytesRx.average",

    "net.bytesTx.average",
```

```
 "net.droppedRx.summation",
    "net.droppedTx.summation",
    "net.usage.average",
    "power.power.average",
    "virtualDisk.numberReadAveraged.average",
    "virtualDisk.numberWriteAveraged.average",
    "virtualDisk.read.average",
    "virtualDisk.readOIO.latest",
    "virtualDisk.throughput.usage.average",
    "virtualDisk.totalReadLatency.average",
    "virtualDisk.totalWriteLatency.average",
    "virtualDisk.write.average",
    "virtualDisk.writeOIO.latest",
    "sys.uptime.latest",
  ]
  # vm_metric_exclude = [] ## Nothing is excluded by default
  # vm_instances = true ## true by default


  ## Hosts
  ## Typical host metrics (if omitted or empty, all metrics are
collected)
  # host_include = [ "/*/host/**"] # Inventory path to hosts to
collect (by default all are collected)
  # host_exclude [] # Inventory paths to exclude
  host_metric_include = [
    "cpu.coreUtilization.average",
    "cpu.costop.summation",
    "cpu.demand.average",
```

```
 "cpu.idle.summation",
    "cpu.latency.average",
    "cpu.readiness.average",
    "cpu.ready.summation",
    "cpu.swapwait.summation",
    "cpu.usage.average",
    "cpu.usagemhz.average",
    "cpu.used.summation",
    "cpu.utilization.average",
    "cpu.wait.summation",
    "disk.deviceReadLatency.average",
    "disk.deviceWriteLatency.average",
    "disk.kernelReadLatency.average",
    "disk.kernelWriteLatency.average",
    "disk.numberReadAveraged.average",
    "disk.numberWriteAveraged.average",
    "disk.read.average",
    "disk.totalReadLatency.average",
    "disk.totalWriteLatency.average",
    "disk.write.average",
    "mem.active.average",
    "mem.latency.average",
    "mem.state.latest",
    "mem.swapin.average",
    "mem.swapinRate.average",
    "mem.swapout.average",
    "mem.swapoutRate.average",
    "mem.totalCapacity.average",
```

```
 "mem.usage.average",
    "mem.vmmemctl.average",
    "net.bytesRx.average",
    "net.bytesTx.average",
    "net.droppedRx.summation",
    "net.droppedTx.summation",
    "net.errorsRx.summation",
    "net.errorsTx.summation",
    "net.usage.average",
    "power.power.average",
    "storageAdapter.numberReadAveraged.average",
    "storageAdapter.numberWriteAveraged.average",
    "storageAdapter.read.average",
    "storageAdapter.write.average",
    "sys.uptime.latest",
  ]
    ## Collect IP addresses? Valid values are "ipv4" and "ipv6"
  # ip_addresses = ["ipv6", "ipv4" ]

  # host_metric_exclude = [] ## Nothing excluded by default
  # host_instances = true ## true by default



  ## Clusters
  # cluster_include = [ "/*/host/**"] # Inventory path to clusters
to collect (by default all are collected)
  # cluster_exclude = [] # Inventory paths to exclude
```

```
# cluster_metric_include = [] ## if omitted or empty, all metrics
are collected

  # cluster_metric_exclude = [] ## Nothing excluded by default

  # cluster_instances = false ## false by default


  ## Resource Pools

  # datastore_include = [ "/*/host/**"] # Inventory path to
datastores to collect (by default all are collected)

  # datastore_exclude = [] # Inventory paths to exclude

  # datastore_metric_include = [] ## if omitted or empty, all
metrics are collected

  # datastore_metric_exclude = [] ## Nothing excluded by default

  # datastore_instances = false ## false by default

  ## Datastores

  # datastore_include = [ "/*/datastore/**"] # Inventory path to
datastores to collect (by default all are collected)

  # datastore_exclude = [] # Inventory paths to exclude

  # datastore_metric_include = [] ## if omitted or empty, all
metrics are collected

  # datastore_metric_exclude = [] ## Nothing excluded by default

  # datastore_instances = false ## false by default

  ## Datacenters

  # datacenter_include = [ "/*/host/**"] # Inventory path to
clusters to collect (by default all are collected)

  # datacenter_exclude = [] # Inventory paths to exclude

  datacenter_metric_include = [] ## if omitted or empty, all
metrics are collected

  datacenter_metric_exclude = [ "*" ] ## Datacenters are not
collected by default.

  # datacenter_instances = false ## false by default
```

```
  ## Plugin Settings

  ## separator character to use for measurement and field names
(default: "_")

  # separator = "_"


  ## number of objects to retrieve per query for realtime resources
(vms and hosts)

  ## set to 64 for vCenter 5.5 and 6.0 (default: 256)

  # max_query_objects = 256


  ## number of metrics to retrieve per query for non-realtime
resources (clusters and datastores)

  ## set to 64 for vCenter 5.5 and 6.0 (default: 256)

  # max_query_metrics = 256


  ## number of go routines to use for collection and discovery of
objects and metrics

  # collect_concurrency = 1

  # discover_concurrency = 1


  ## the interval before (re)discovering objects subject to metrics
collection (default: 300s)

  # object_discovery_interval = "300s"


  ## timeout applies to any of the api request made to vcenter

  # timeout = "60s"


  ## When set to true, all samples are sent as integers. This makes
the output
```

```
## data types backwards compatible with Telegraf 1.9 or lower.
Normally all

  ## samples from vCenter, with the exception of percentages, are
integer

  ## values, but under some conditions, some averaging takes place
internally in

  ## the plugin. Setting this flag to "false" will send values as
floats to

  ## preserve the full precision when averaging takes place.

  # use_int_samples = true


  ## Custom attributes from vCenter can be very useful for queries
in order to slice the

  ## metrics along different dimension and for forming ad-hoc
relationships. They are disabled

  ## by default, since they can add a considerable amount of tags
to the resulting metrics. To

  ## enable, simply set custom_attribute_exclude to [] (empty set)
and use custom_attribute_include

  ## to select the attributes you want to include.

  ## By default, since they can add a considerable amount of tags
to the resulting metrics. To

  ## enable, simply set custom_attribute_exclude to [] (empty set)
and use custom_attribute_include

  ## to select the attributes you want to include.

  # custom_attribute_include = []

  # custom_attribute_exclude = ["*"]


  ## The number of vSphere 5 minute metric collection cycles to
look back for non-realtime metrics. In

  ## some versions (6.7, 7.0 and possible more), certain metrics,
such as cluster metrics, may be reported
```

```
## with a significant delay (>30min). If this happens, try
increasing this number. Please note that increasing

  ## it too much may cause performance issues.

  # metric_lookback = 3


  ## Optional SSL Config

  # ssl_ca = "/path/to/cafile"

  # ssl_cert = "/path/to/certfile"

  # ssl_key = "/path/to/keyfile"

  ## Use SSL but skip chain & host verification

  # insecure_skip_verify = false


  ## The Historical Interval value must match EXACTLY the interval
in the daily

  # "Interval Duration" found on the VCenter server under Configure
> General > Statistics > Statistic intervals

  # historical_interval = "5m"
```

Додаток В

Специфікація розгортання стеку моніторингу в Docker Compose.

```
services:
  influxdb:
    image: influxdb:2.7-alpine
    container_name: influxdb
    restart: always
    volumes:
      - influxdbv2:/var/lib/influxdb2:rw
    ports:
      - "8086:8086"
    networks:
      - monitoring  # В одній мережі
  nginx:
    image: nginx:alpine
    container_name: nginx
    restart: always
    volumes:
      - ./nginx:/etc/nginx/conf.d:ro  # Монтуємо як додаток до конфігу
    ports:
      - '8080:80'
    networks:
      - monitoring
    depends_on:
      - influxdb  # Запуск після бази
  grafana:
    image: grafana/grafana-enterprise
```

```
container_name: grafana
    restart: always
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=admin
    ports:
      - '3000:3000'
    volumes:
      - grafana_data:/var/lib/grafana
    networks:
      - monitoring
  telegraf:
    image: telegraf:1.25-alpine
    container_name: telegraf
    restart: always
    volumes:
      - ./telegraf/mytelegraf.conf:/etc/telegraf/telegraf.conf:ro
    networks:
      - monitoring
    depends_on:
      - influxdb
volumes:
  influxdbv2:
  grafana_data:
networks:
  monitoring:
    driver: bridge
```

Додаток Г

Текст запиту для розрахунку та візуалізації вільного місця на дискових масивах.

```
// 1. Отримуємо дані про загальний об'єм сховища (capacity)
capacity = from(bucket: "vsphere_telemetry")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "capacity_latest")
  |> last()

// 2. Отримуємо дані про вільний об'єм (free space)
free = from(bucket: "vsphere_telemetry")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "free_space_latest")
  |> last()

// 3. Об'єднуємо два потоки даних та проводимо розрахунок відсотків
join(tables: {cap: capacity, fre: free}, on: ["datastore",
"esxhost"])
  |> map(fn: (r) => ({
      _time: r._time_cap,
      datastore: r.datastore,
      // Формула: (Вільний об'єм / Загальний об'єм) * 100

     percent_free: float(v: r._value_fre) / float(v: r._value_cap)
* 100.0
    }))
  |> yield(name: "storage_free_percentage")
```

Додаток Д

Лістинг аналітичного запиту мовою Flux для вибірки даних із InfluxDB.

```
// 1. Отримуємо дані про загальний об'єм сховища (capacity)
capacity = from(bucket: "vsphere_telemetry")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "capacity_latest")
  |> last()

// 2. Отримуємо дані про вільний об'єм (free space)
free = from(bucket: "vsphere_telemetry")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r["_measurement"] == "vsphere_datastore")
  |> filter(fn: (r) => r["_field"] == "free_space_latest")
  |> last()

// 3. Об'єднуємо два потоки даних та проводимо розрахунок відсотків
join(tables: {cap: capacity, fre: free}, on: ["datastore",
"esxhost"])
  |> map(fn: (r) => ({
      _time: r._time_cap,
      datastore: r.datastore,
      // Формула: (Вільний об'єм / Загальний об'єм) * 100

     percent_free: float(v: r._value_fre) / float(v: r._value_cap)
* 100.0
    }))
  |> yield(name: "storage_free_percentage")
```