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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «СИСТЕМА КЕРУВАННЯ ВІРТУАЛЬНИМИ СЕРЕДОВИЩАМИ ТА СЛУЖБАМИ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

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

_____ *Антон ХОДОСОВ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КСДМ-62
Антон ХОДОСОВ
Ім'я, ПРІЗВИЩЕ
Керівник: Наталія ЛАЩЕВСЬКА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії
Ступінь вищої освіти Магістр
Спеціальність 123 Комп'ютерної інженерії
Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
Ім'я, ПРІЗВИЩЕ
«_____» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ходосов Антон Олександрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система керування віртуальними середовищами та службами
керівник кваліфікаційної роботи Наталія ЛАЩЕВСЬКА,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» 10 2025р. № 467
2. Строк подання кваліфікаційної роботи «_____» _____ 2025р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, методи порівняльного аналізу, метод розгортання та конфігурування програмних комплексів, а також емпіричні методи для перевірки працездатності системи.
Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи моніторингу серверної інфраструктури
2. Архітектура та функціонал системи zabbix
3. Практична реалізація моніторингу на базі zabbix
4. Перелік ілюстративного матеріалу: *презентація*
5. Дата видачі завдання «_____» _____ 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.09-24.10.2025	Виконано
2	Обробка матеріалу	25.10-05.11.2025	Виконано
3	Розробка теоретичної частини	06.11-10.11.2025	Виконано
4	Розробка системи	11.11-15.11.2025	Виконано
5	Висновки за результатами аналізу	15.11-19.11.2025	Виконано
6	Оформлення пояснювальної записки	19.11-21.11.2025	Виконано
7	Розробка презентації	21.11-29.11.2025	Виконано
8	Передзахист	01.12-02.12.2025	Виконано
9	Здача в деканат	15.12-18.12.2025	Виконано

Здобувач вищої освіти _____ Антон ХОДОСОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Наталія ЛАЩЕВСЬКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 65 стор., 43 рис., 2 табл., 18 джерел.

Мета дослідження – підвищення надійності та ефективності функціонування IT-інфраструктури шляхом розробки та впровадження автоматизованої системи моніторингу віртуальних середовищ та служб на базі платформи Zabbix 7.0 LTS.

Об'єкт дослідження – процеси адміністрування та технічного нагляду за станом серверного обладнання та програмного забезпечення у віртуалізованих системах.

Предмет дослідження – методи та інструментальні засоби збору, аналізу та візуалізації метрик продуктивності серверів з використанням агентних технологій та протоколів передачі даних.

Короткий зміст роботи: У кваліфікаційній роботі проведено порівняльний аналіз сучасних систем моніторингу та обґрунтовано вибір Zabbix як оптимального Open Source рішення. Розроблено архітектуру системи моніторингу, що включає серверну частину на базі Ubuntu 24.04 LTS, веб-сервер Nginx та СУБД MySQL. Реалізовано розгортання Zabbix Agent 2 для збору метрик у реальному часі. Налаштовано спеціалізований моніторинг баз даних із використанням макросів та принципів мінімальних привілеїв. Проведено верифікацію системи шляхом стрес-тестування (синтетичне навантаження), що підтвердило коректність спрацювання тригерів та сповіщень.

Результати: Розроблена система забезпечує централізований контроль апаратних ресурсів та прикладних служб, дозволяє виявляти деградацію продуктивності на ранніх етапах та автоматизувати реакцію на інциденти, що мінімізує час простою сервісів.

КЛЮЧОВІ СЛОВА: МОНІТОРИНГ, ВІРТУАЛІЗАЦІЯ, ZABBIX, UBUNTU, MYSQL, NGINX, АГЕНТ, ALERTING, СТРЕС-ТЕСТУВАННЯ, АДМІНІСТРУВАННЯ.

ABSTRACT

Textual part of the Master's thesis: 65 pages, 43 figures, 2 tables, 18 references.

The aim of the study is to increase the reliability and efficiency of IT infrastructure functioning by developing and implementing an automated monitoring system for virtual environments and services based on the Zabbix 7.0 LTS platform.

The object of the study is the processes of administration and technical supervision of the state of server hardware and software in virtualized systems.

The subject of the study is the methods and tools for collecting, analyzing, and visualizing server performance metrics using agent technologies and data transfer protocols.

Content summary: The thesis conducts a comparative analysis of modern monitoring systems and justifies the choice of Zabbix as the optimal Open Source solution. The architecture of the monitoring system has been developed, including a server part based on Ubuntu 24.04 LTS, Nginx web server, and MySQL DBMS. The deployment of Zabbix Agent 2 for real-time metric collection has been implemented. Specialized database monitoring has been configured using macros and least privilege principles. System verification was performed via stress testing (synthetic load), confirming the correct operation of triggers and notifications.

Results: The developed system ensures centralized control of hardware resources and application services, enables early detection of performance degradation, and automates incident response, thereby minimizing service downtime.

KEYWORDS: MONITORING, VIRTUALIZATION, ZABBIX, UBUNTU, MYSQL, NGINX, AGENT, ALERTING, STRESS TESTING, ADMINISTRATION.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Ходосов А. О. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 123 Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)
на тему: «Система керування віртуальними середовищами та службами».
Кваліфікаційна робота і рецензія додаються.

Директор ННІ _____ Катерина НЕСТЕРЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Ходосов А. О. показав відмінну теоретичну підготовку, уміння володіти та розуміння новими комп'ютерними технологіями, користуватися навчальною і науково-технічною літературою. Працюючи над завданнями, які були поставлені керівником, проявив ініціативність, сумлінність та хист до інженерної роботи.
Все це дозволяє оцінити кваліфікаційну роботу студента Ходосова Антона Олександровича на оцінку «відмінно», та присвоїти їй кваліфікацію «магістр з комп'ютерної інженерії».

Керівник кваліфікаційної роботи _____ Наталія ЛАЩЕВСЬКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«__» _____ 2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач *Ходосов А. О.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Комп'ютерної інженерії _____ Наталія ЛАЩЕВСЬКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну магістерську роботу

здобувача вищої освіти Ходосов Антон Олександрович
(прізвище, ім'я, по батькові)
на тему: «Система керування віртуальними середовищами та службами»

Актуальність:

Кваліфікаційна магістерська робота присвячена розробці та впровадженню автоматизованої системи моніторингу IT-інфраструктури на базі платформи Zabbix. Тема є вкрай актуальною в умовах масового переходу підприємств на віртуалізовані середовища, де критично важливим є забезпечення безперервності бізнес-процесів. Робота пропонує практичний підхід до вирішення проблеми проактивного виявлення збоїв, контролю використання ресурсів та аналізу продуктивності служб, що дозволяє мінімізувати час простою систем та оптимізувати роботу адміністраторів.

Позитивні сторони:
1. У роботі проведено глибокий порівняльний аналіз сучасних систем моніторингу, що дозволило аргументовано обґрунтувати вибір Zabbix 7.0 LTS як найбільш ефективного Open Source рішення для поставлених завдань .
2. Автор продемонстрував високий рівень технічної підготовки, реалізувавши повний цикл розгортання складної архітектури: від підготовки серверної ОС Ubuntu та налаштування веб-сервера Nginx до оптимізації СУБД MySQL та конфігурації Zabbix Agent 2 .
3. Особливої цінності набуває практична частина, присвячена поглибленому моніторингу баз даних із використанням користувацьких макросів та проведенню стрес-тестування за допомогою утиліти mysqlslap. Це дозволило на практиці перевірити реакцію системи на пікові навантаження та коректність роботи тригерів .

Недоліки:
1. В роботі основну увагу приділено налаштуванню серверної частини та збору метрик, проте недостатньо детально розкрито питання налаштування захищеного з'єднання для веб-інтерфейсу Zabbix, що є важливим аспектом безпеки, хоча теоретичні основи шифрування були розглянуті .
2. Бажано було б детальніше розглянути налаштування каналів сповіщення у практичній частині, оскільки робота демонструє фіксацію інцидентів переважно через внутрішній інтерфейс без опису інтеграції із зовнішніми месенджерами .

Висновок: *кваліфікаційна бакалаврська робота заслуговує оцінку " відмінно", а здобувач Ходосов А. О. заслуговує присвоєння кваліфікації: магістр з комп'ютерної інженерії.*

Рецензент: _____ _____
науковий ступінь, вчене звання *підпис* *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП.....9
1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ.....11
1.1 Сутність та актуальність нагляду за серверами.....11
1.2 Виклики в управлінні IT-інфраструктурою.....12
1.3 Специфіка віртуалізації.....16
1.4 Інструментарій керування та класифікація систем.....17
2 АРХІТЕКТУРА ТА ФУНКЦІОНАЛ СИСТЕМИ ZABBIX.....23
2.1 Призначення Zabbix.....23
2.2 Компоненти архітектури.....24
2.3 Механізми збору та захисту даних.....25
2.4 Розширені можливості: Бази даних та Веб-хуки.....27
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОНІТОРИНГУ НА БАЗІ ZABBIX.....30
3.1 Підготовка серверного середовища та розгортання репозиторіїв.....30
3.2 Розгортання бази даних.....33
3.3 Налаштування Firewall, Nginx та оркестрація служб.....36
3.4 Ініціалізація веб-інтерфейсу та фінальна конфігурація Zabbix Frontend.....39
3.5 Розгортання агента моніторингу Zabbix Agent 2 та реєстрація цільового хоста.....43
3.6 Організація прикладного моніторингу СУБД MySQL.....48
3.7 Верифікація системи моніторингу шляхом генерації синтетичного навантаження.....51
3.8 Діагностика аварійних станів та аналіз механізмів сповіщення про інциденти.....57
ВИСНОВКИ.....60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....62
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація).....65

ВСТУП

У сучасному інформаційному просторі серверна інфраструктура є фундаментом для функціонування будь-якого бізнесу. Акцент змістився з фізичних серверів на віртуальні середовища, які можуть одночасно експлуатуватися тисячами користувачів. У таких умовах забезпечення безперервної експлуатації, високої доступності та відмовостійкості сервісів стає критично важливим завданням. Вихід з ладу навіть одного елемента системи може призвести не лише до фінансових втрат, але й до зупинки критично важливих бізнес-процесів.

Керування сучасною гетерогенною інфраструктурою вручну є неефективним та неможливим через великий обсяг даних. Типові «коробкові» рішення часто не надають повної картини, а затримка у виявленні несправностей призводить до тривалих простоїв. Це зумовлює необхідність впровадження спеціалізованих автоматизованих систем моніторингу, здатних не лише фіксувати збої, але й прогнозувати їх на основі аналізу історичних даних.

Мета і завдання дослідження. Метою роботи є побудова комплексної системи керування та моніторингу віртуальними середовищами, яка забезпечує проактивне виявлення проблем та аналіз продуктивності служб.

Для досягнення мети поставлені наступні завдання:

1. Проаналізувати теоретичні основи моніторингу та порівняти існуючі програмні рішення.
2. Обґрунтувати вибір програмного стеку для реалізації системи .
3. Розгорнути та налаштувати серверну частину системи моніторингу, включаючи конфігурацію бази даних та веб-інтерфейсу.
4. Впровадити агентний моніторинг для збору метрик з віртуальних хостів.

5. Реалізувати поглиблений моніторинг прикладного рівня з налаштуванням користувацьких макросів.

6. Провести експериментальну перевірку ефективності системи шляхом генерації штучного навантаження та аналізу реакції підсистеми сповіщень.

Об’єкт дослідження – процеси функціонування віртуальних серверних середовищ. Предмет дослідження – методи та засоби автоматизованого моніторингу стану IT-інфраструктури на базі Zabbix.

Методи дослідження. У роботі використано методи порівняльного аналізу, метод розгортання та конфігурування програмних комплексів, а також емпіричні методи для перевірки працездатності системи.

Практичне значення. Розроблене рішення дозволяє системним адміністраторам отримувати повну видимість стану інфраструктури в режимі реального часу, що є основою для забезпечення стабільності сервісів та оптимізації використання ресурсів.

1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

1.1 Сутність та актуальність нагляду за серверами

Формулювання універсального визначення поняття «моніторинг серверів» є складним завданням через значну гетерогенність сучасної серверної інфраструктури. Якщо раніше під сервером розуміли виключно фізичний пристрій, то сьогодні акцент змістився на віртуальні сервери, що функціонують на базі фізичних хостів у масштабних центрах обробки даних. Такі ресурси можуть одночасно експлуатуватися тисячами незалежних груп клієнтів, кожна з яких адмініструє власні системи. Функціональне призначення серверів варіюється від поштових служб, веб-серверів та серверів друку до баз даних і архітектури мікросервісів, що працюють під управлінням різноманітних операційних систем та програмних продуктів[1].

Сам процес моніторингу являє собою механізм отримання та аналізу даних про активність серверів (як віртуальних, так і фізичних) для забезпечення прозорості їх функціонування. Враховуючи, що сервери обробляють тисячі запитів на отримання та передачу даних, вихід з ладу навіть однієї одиниці обладнання може дестабілізувати роботу критично важливих систем. Наслідки таких інцидентів виходять за межі втрати часу чи фінансів компанії; у критичних сферах це може загрожувати майну, здоров’ю або навіть життю людей. Саме тому забезпечення відмовостійкості апаратно-програмного комплексу є пріоритетним завданням фахівців з управління IT-інфраструктурою.

Ефективний нагляд за станом серверів вимагає спеціалізованих технологічних рішень, оскільки типові «коробкові» інструменти моніторингу часто не здатні надати повну картину процесів, що відбуваються в системі. У

такій ситуації інженери постають перед вибором: або модифікувати готовий програмний продукт під специфічні потреби (що пов’язано з ризиками некоректної роботи та складнощами реалізації), або розгортати мережу з багатьох розрізнених сервісів моніторингу, кожен з яких потребує окремого супроводу та контролю.

1.2 Виклики в управлінні IT-інфраструктурою

Управління сервером визначається як процес забезпечення його безперервної експлуатації, що вимагає постійного контролю за надійністю, продуктивністю, доступністю та безпомилковістю функціонування систем. Це щоденна операційна діяльність, пріоритетом якої є гарантування стабільності сервісів для ефективної взаємодії з кінцевим користувачем[2].

Спектр завдань адміністрування варіюється залежно від масштабу інфраструктури та типу організації. Типовий набір функцій при роботі з віртуальними машинами включає щоденний моніторинг, інсталяцію та конфігурацію нових компонентів, оновлення програмного та апаратного забезпечення, а також діагностику та превентивне усунення несправностей.

Окремим викликом є балансування навантаження на IT-сервіси. Критичні ситуації можуть виникати, наприклад, під час масового формування звітності, коли запити користувачів різко зростають. Завдання фахівця — передбачити такі пікові моменти та забезпечити необхідні потужності. Водночас постійне утримання надлишкових ресурсів «про запас» є економічно недоцільним і призводить до фінансових втрат, що може викликати невдоволення замовника. Ефективне вирішення цієї дилеми можливе лише шляхом аналізу історичних даних, зібраних у попередні періоди[3].

Специфічні труднощі виникають при роботі з віртуальними середовищами, де адміністратор часто позбавлений можливості фізичної

перевірки обладнання, оскільки сервери можуть знаходитися у віддалених дата-центрах з обмеженим доступом. Водночас наявність прямого фізичного доступу до серверів породжує інший клас експлуатаційних проблем. Незалежно від типу розгортання, критично важливим залишається комплексне управління апаратними та програмними ресурсами, контроль пропускної здатності мережі, а також забезпечення стабільного електропостачання та охолодження[4].

1.3 Аналіз сучасних підходів до організації моніторингу локальних мереж

Моніторинг мережі визначається як процес систематичного контролю за станом комп'ютерної інфраструктури, спрямований на виявлення компонентів із низькою продуктивністю або несправностями. Головна мета цього процесу — запобігання зниженню швидкості передачі даних та деградації пропускної здатності каналів зв'язку. Об'єктами спостереження виступають сервери (які можуть бути перевантаженими або «завислими»), маршрутизатори, комутатори та інше активне мережеве обладнання. У разі виникнення критичних помилок або повного виходу вузла з ладу система автоматично сповіщає адміністратора. Варто зазначити, що моніторинг є складовою частиною ширшої дисципліни — управління мережею[5].

Для реалізації завдань моніторингу використовується широкий спектр засобів: від базових діагностичних утиліт, таких як Ping, до комплексних систем аналізу трафіку, що здатні відстежувати специфічні потоки даних (відеоконтент, VoIP, поштові протоколи POP3). Застосування спеціалізованих апаратних та програмних агентів дозволяє досягти повної прозорості (visibility) інфраструктури, контролюючи кожен підключений пристрій[6].

Ключові переваги та функції сучасних систем мережевого моніторингу включають:

— Забезпечення нормативної відповідності (Compliance). Інструменти моніторингу є обов'язковим елементом внутрішнього контролю для організацій, що підпадають під дію регуляторних стандартів (PCI DSS, HIPAA, FISMA, SOX). Це дозволяє доповнити зовнішні периметри безпеки внутрішнім наглядом.

— Мінімізація простоїв (Downtime prevention). Оскільки перерви в роботі сервісів призводять до значних фінансових втрат, системи моніторингу фокусуються на виявленні «симптомів» майбутніх збоїв обладнання. Це дозволяє адміністраторам діяти на випередження, проводячи обслуговування до моменту аварійної зупинки.

— Прискорення діагностики та усунення несправностей. Завдяки автоматизованій побудові карт мережі та аналітичним інструментам, фахівці можуть швидко локалізувати джерело проблеми — чи то сплеск трафіку, чи помилка в конфігурації пристрою.

— Виявлення загроз інформаційній безпеці. Окрім контролю продуктивності, системи моніторингу виконують роль інструменту безпеки, фіксуючи аномальну активність або підозрілі патерни поведінки, що може свідчити про кібератаку або несанкціонований доступ.

Ідентифікація деградації продуктивності на ранніх етапах є фундаментальним елементом системи мережевого моніторингу[7]. Застосування ефективних превентивних стратегій дозволяє уникнути аварійних простоїв та критичних збоїв у роботі інфраструктури. У контексті даного дослідження моніторинг відіграє ключову роль, оскільки об'єктивна оцінка ефективності локальної обчислювальної мережі (ЛОМ) неможлива без аналізу вичерпного масиву статистичних даних. Для формування цілісної картини функціонування мережі критично важливим є контроль таких метрик, як пропускна здатність каналів, час затримок (latency) та рівень навантаження на обладнання.

З технічної точки зору, моніторинг ЛОМ можна визначити як систематичний процес збору та акумуляції телеметричних даних, що реалізується за допомогою стандартних протоколів керування мережею. Система отримує від мережевих вузлів інформацію про швидкодію, утилізацію каналів зв'язку, затримки пакетів та інші параметри. Надалі цей масив даних підлягає аналітичній обробці, на основі якої формуються деталізовані звіти у форматі, зручному для сприйняття та прийняття управлінських рішень.

Протоколи керування мережею визначаються як стандартизовані набори правил та процедур, що регламентують взаємодію мережевих пристроїв та забезпечують механізми контролю за комунікаційними процесами. Функціональне навантаження цих протоколів охоплює широкий спектр завдань, включаючи маршрутизацію трафіку, комутацію пакетів, логічну адресацію та управління доступом до середовища передачі даних[8].

До переліку ключових протоколів, що забезпечують керованість та стабільність мережі, належать:

— RIP (Routing Information Protocol): Протокол маршрутизації, призначений для обміну інформацією про топологію мережі та доступні маршрути між шлюзами.

— SNMP (Simple Network Management Protocol): Фундаментальний стандарт для збору телеметрії про стан апаратного та програмного забезпечення мережевих вузлів. Він надає адміністраторам інструменти для централізованого моніторингу та віддаленої конфігурації обладнання.

— DHCP (Dynamic Host Configuration Protocol): Протокол, що автоматизує процес призначення мережевих параметрів (зокрема IP-адрес, масок підмережі, шлюзів) клієнтським пристроям, спрощуючи адміністрування.

— VRRP (Virtual Router Redundancy Protocol): Механізм забезпечення високої доступності та відмовостійкості, який дозволяє групі маршрутизаторів

використовувати спільну віртуальну IP-адресу, гарантуючи безперервність роботи шлюзу при збоях фізичних пристроїв.

— OSPF (Open Shortest Path First): Протокол динамічної маршрутизації на основі стану каналів, який обчислює найкоротші та найбільш ефективні шляхи передачі даних всередині автономної системи.

— ICMP (Internet Control Message Protocol): Службовий протокол для діагностики мережевих з’єднань (зокрема за допомогою утиліти ping) та передачі службових повідомлень про помилки при доставці пакетів.

Інтегроване використання цих протоколів є необхідною умовою для побудови надійної комп'ютерної мережі, забезпечуючи ефективну комунікацію між пристроями та стабільний обмін інформаційними потоками.

1.4 Специфіка віртуалізації

Віртуальні сервери функціонують як програмна емуляція апаратних компонентів серверного обладнання. Поширення цієї технології було зумовлене необхідністю оптимізації використання обчислювальних ресурсів, оскільки системні адміністратори виявили проблему неповного завантаження потужностей фізичних машин. У ситуаціях, коли організація задіює лише частину свого обчислювального потенціалу, виникає надлишок невикористаних ресурсів, які простоюють. Враховуючи високу вартість технічного обслуговування, адміністрування та забезпечення безпеки фізичного обладнання, міграція у віртуальне середовище є обґрунтованою з позиції рентабельності інвестицій (ROI).

Надання доступу до віртуальних потужностей зазвичай здійснюється спеціалізованими сертифікованими провайдерами, які управляють масштабною інфраструктурою, що налічує десятки або сотні тисяч фізичних хостів у глобальних дата-центрах. Така модель дозволяє гнучко орендувати,

ініціалізувати та керувати ресурсами, забезпечуючи адміністраторам можливість динамічного масштабування інфраструктури відповідно до раптових сплесків або зниження навантаження . Економічна ефективність віртуальних серверів додатково підсилюється моделлю оплати, яка у більшості провайдерів базується на фактичному споживанні ресурсів, що робить їх використання значно вигіднішим порівняно з експлуатацією фізичних серверів[9].

1.5 Інструментарій керування та класифікація систем

Системи управління серверами — це спеціалізоване програмне забезпечення, призначене для адміністрування серверних кластерів. Ключовою функцією таких систем є моніторинг, що передбачає збір оперативних метрик, серед яких: навантаження на центральний процесор (CPU), використання дискового простору та оперативної пам’яті (RAM), пропускна здатність мережевих інтерфейсів, кількість активних підключень, а також дані щодо вразливостей та сповіщення від систем безпеки.

Зібрана інформація візуалізується на інформаційних панелях (dashboards) у режимі реального часу, а також архівується для ретроспективного аналізу, що дозволяє фахівцям відстежувати динаміку змін показників у часі . У контексті віртуалізації важливо розрізняти системи моніторингу та гіпервізори (монітори віртуальних машин). Якщо функцією гіпервізора є створення та управління життєвим циклом віртуальних машин згідно зі специфікаціями, то системи моніторингу фокусуються саме на аналізі їхньої продуктивності.

Існує кілька підходів до класифікації систем моніторингу[10]:

— Локальні (On-premise): Традиційні рішення, що розгортаються у внутрішньому периметрі.

— Хмарні (SaaS): Сервіси, що надаються та керуються через інтернет.

— Гібридні та мобільні: Поєднують характеристики локальних та хмарних технологій.

Особливості SaaS-моніторингу: Модель «Програмне забезпечення як послуга» (SaaS) дозволяє уникнути складної інсталяції в інфраструктурі користувача, забезпечуючи швидке (протягом кількох годин) віддалене налаштування. Хоча такі системи можуть надавати менший рівень прямого контролю над конфігурацією, вони компенсують це високою гнучкістю та низьким порогом входження. Конкуренція на ринку SaaS стимулює постійний розвиток продуктів, а модель передплати без довгострокових зобов'язань знижує експлуатаційні ризики для замовника.

При виборі оптимального рішення для контролю стану серверів фахівцям необхідно спиратися на наступні ключові критерії:

— Масштабованість та універсальність: Інструмент повинен підтримувати всі типи наявного обладнання (локальні, хмарні сервери) та бути придатним для використання після майбутніх модернізацій інфраструктури.

— Гнучкість системи сповіщень: Можливість кастомізації формату повідомлень та підтримка зручних каналів доставки сповіщень для цільових груп фахівців.

— Аналітика першопричин (Root Cause Analysis): Наявність інтелектуальних алгоритмів (включно зі ШІ), які не просто фіксують помилку, а надають контекст та допомагають визначити причину інциденту.

— Ергономіка та юзабіліті: Інтуїтивно зрозумілий інтерфейс, що спрощує сортування подій та пришвидшує реакцію на них.

— Якість технічної підтримки: Політика взаємодії вендора з користувачами та доступність кваліфікованої допомоги.

1.6 Аналіз та порівняльна характеристика систем моніторингу мережевої інфраструктури

Забезпечення стабільності комп'ютерної мережі вимагає безперервного контролю її стану. Систематичний моніторинг дозволяє адміністраторам оперативно виявляти загрози та запобігати збоям у функціонуванні інфраструктури.

У контексті діагностики важливо розрізняти програмні мережеві монітори та апаратні аналізатори.

Мережеві монітори: Програмні засоби, що збирають статистичні дані про трафік (середня інтенсивність, потік пакетів, статистика помилок тощо).

Мережеві аналізатори: Спеціалізовані дороговартісні (зазвичай понад $20 000) апаратні комплекси для лабораторного аналізу фізичних характеристик кабельних ліній та електромагнітних параметрів.

Для цілей даної роботи доцільно розглянути програмні рішення. Нижче наведено огляд трьох актуальних систем.

Network Olympus. Інноваційне рішення, що працює у форматі системної служби та керується через веб-інтерфейс.

— Ключова особливість: Гнучкий конструктор сценаріїв, що базується на сенсорах. Це дозволяє створювати складні логічні ланцюжки перевірок замість примітивних тестів.

— Функціонал: Залежно від реакції сенсора, система може ініціювати різноманітні дії або сповіщення для автоматизації усунення проблем.

— Переваги: Можливість редагування логіки роботи «на льоту» (зміни миттєво застосовуються до всіх пристроїв) та детальне протоколювання подій у журналах.

Observium. Платформа, орієнтована на глибокий аналіз продуктивності та стану мережі, що базується на протоколі SNMP.

— Сумісність: Підтримує широкий спектр обладнання від провідних вендорів (Cisco, Juniper, Dell, HP, NetApp та ін.) та операційні системи Windows, Linux, FreeBSD.

— Візуалізація: Система вирізняється якісним графічним інтерфейсом, який автоматично будує діаграми та графіки, наочно підсвічуючи «вузькі місця» інфраструктури.

— Ліцензування: Доступна у двох версіях — обмеженій Community (демо) та повнофункціональній платній версії (Professional), вартість якої становить близько 200 фунтів стерлінгів на рік.

Zabbix. Універсальне Open Source рішення корпоративного рівня, спроектоване для моніторингу масштабних гетерогенних середовищ [6].

— Архітектура: Підтримує багатосерверні інфраструктури та роботу з різними ОС (Linux, Windows, FreeBSD). Здатна ефективно керувати сотнями вузлів.

— Методи збору даних: Використовує власні програмні агенти, SNMP або інші протоколи віддаленого доступу. Управління здійснюється через PHP-веб-інтерфейс[11].

— Вимоги: Для реалізації складних сценаріїв та глибокої кастомізації адміністратору можуть знадобитися навички скриптування (Python, Perl).

За результатами порівняльного аналізу, табл.1.1, найбільш оптимальним рішенням для реалізації поставлених завдань визначено систему Zabbix. Основні аргументи на користь цього вибору:

1. Економічна ефективність: Система є повністю безкоштовною (Open Source), що є критичним для невеликих підприємств.

2. Гнучкість: Підтримка плагінів та широкі можливості масштабування.

3. Автоматизація реакцій: Потужна підсистема сповіщень дозволяє не лише фіксувати інциденти, а й миттєво передавати інформацію в месенджери (Telegram, Slack) або чат-боти, що значно прискорює час реакції на збої[17].

Таблиця 1.1 – Порівняльна характеристика сучасних систем моніторингу

Критерій порівняння	Network Olympus	Observium	Zabbix
Тип рішення	Програма-служба	Платформа на базі SNMP	Універсальне Open Source рішення
Основний метод збору даних	Сенсори та сценарії	Протокол SNMP	Програмні агенти, SNMP, віддалений доступ
Ключові функціональні особливості	Конструктор сценаріїв з логічними ланцюжками; можливість редагування логіки «на льоту»	Глибокий аналіз продуктивності; автовизначення обладнання; детальні технічні характеристики	Підтримка багатосерверної архітектури; потужна система сповіщень (месенджери, боти); підтримка плагінів
Підтримка обладнання/ОС	Універсальна (через сценарії)	Широка інтеграція (Cisco, HP, Juniper, Dell, Windows, Linux, FreeBSD та ін.)	Linux, FreeBSD, Windows (серверна частина та агенти)
Інтерфейс та візуалізація	Зручний веб-інтерфейс; журнали активності та звіти	Розвинений графічний інтерфейс; автоматична побудова діаграм та графіків	Веб-інтерфейс на PHP; гнучкі налаштування дашбордів

Продовження таблиці 1.1

Масштабованість	Орієнтація на автоматизацію процесів	Мережі будь-якого масштабу	Ефективне управління сотнями вузлів; підходить для великих підприємств
Вимоги до кваліфікації	Навички побудови логічних схем	Базові навички адміністрування	Бажані знання скриптових мов (Perl, Python) для повного розкриття функціоналу

2 АРХІТЕКТУРА ТА ФУНКЦІОНАЛ СИСТЕМИ ZABBIX

2.1 Призначення Zabbix

Zabbix позиціонується як комплексний інструмент, що надає системним адміністраторам можливість отримувати актуальні дані про стан різнорідних елементів IT-ландшафту . Сфера його моніторингу охоплює віртуальні сервери, розгорнуте на них програмне забезпечення, а також стан мережевої інфраструктури підприємств чи організацій. Ключовою перевагою архітектури є централізація керування: весь функціонал доступний через уніфіковану веб-консоль, що значно оптимізує повсякденну роботу фахівців. Окрім інфраструктурних компонентів, сервіс також підтримує моніторинг продуктивності веб-сайтів.

Впровадження Zabbix є особливо доцільним у сценаріях, коли необхідний жорсткий контроль ресурсів, але у персоналу відсутній часовий ресурс на ручний аналіз журналів подій (логів) чи безперервний моніторинг екранів у реальному часі. Велика кількість системних повідомлень та логів ускладнює виявлення критичних попереджень про збої. У таких ситуаціях Zabbix бере на себе функцію автоматизованого нагляду, інформуючи адміністратора не лише про фактичні аварії, а й про потенційні проблеми, що можуть виникнути у майбутньому[12].

Робота системи базується на відстеженні метрик та реагуванні на перетин заданих порогових значень (тригерів). Наприклад, система може зафіксувати критичне використання ресурсу, таке як завантаження оперативної пам'яті віртуальної машини понад 90% від загального обсягу. При настанні такої події Zabbix автоматично генерує та надсилає сповіщення через налаштовані канали зв'язку: електронну пошту, SMS-повідомлення або месенджери, забезпечуючи оперативну реакцію персоналу.

2.2 Компоненти архітектури

Функціонування платформи забезпечується взаємодією кількох ключових програмних компонентів, кожен з яких виконує специфічну роль у загальній екосистемі, рис. 2.1.

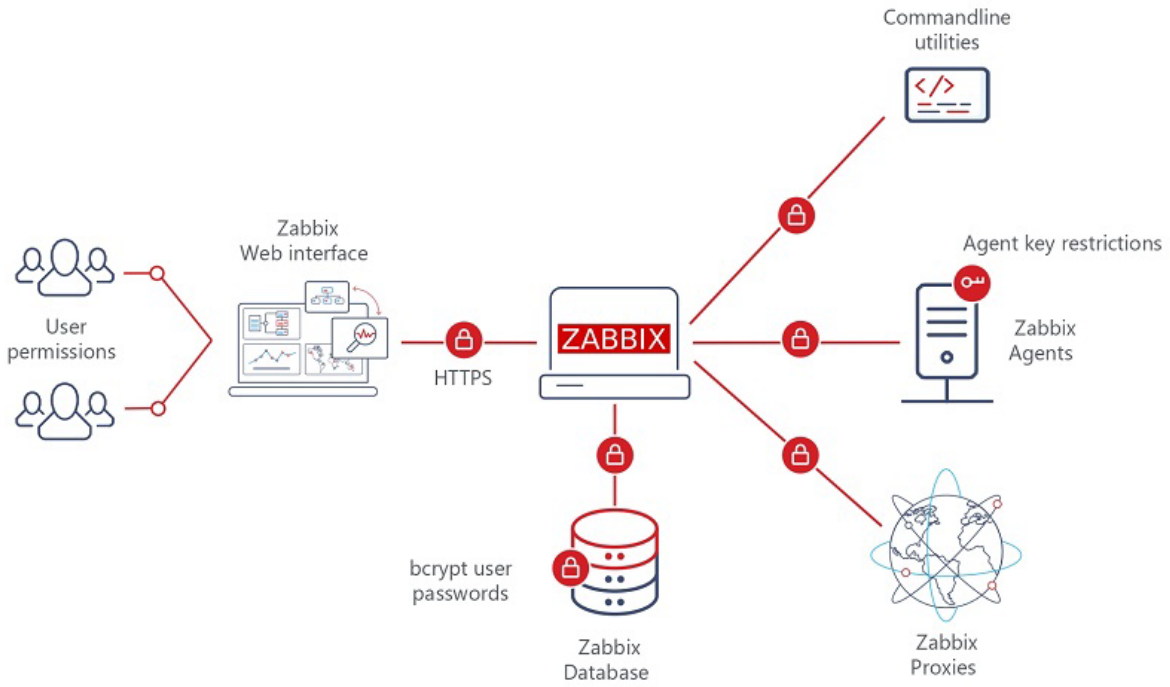


Рисунок 2.1 – Компоненти архітектури Zabbix

Компоненти архітектури Zabbix[18] включають в себе:

— Zabbix-сервер. Це центральний елемент системи, який відповідає за агрегацію даних, що надходять від колекторів (агентів), розміщених на контрольованих хостах. Сервер виконує роль головного репозиторію, де централізовано зберігаються конфігурації, статистична інформація та оперативні дані про роботу сервісів.

— База даних. Як і будь-який програмний комплекс, Zabbix потребує надійного сховища для запису інформації. Система підтримує роботу з широким спектром систем керування базами даних (СКБД), зокрема: MySQL, Oracle, PostgreSQL, SQLite та IBM DB2.

— Web-інтерфейс. Для візуалізації метрик та спрощення процесу налаштування система оснащена інтуїтивно зрозумілим веб-інтерфейсом. Це забезпечує гнучкість доступу до панелі моніторингу з будь-якої локації та пристрою (ПК, планшет, смартфон). Зазвичай інтерфейс функціонує на тій самій фізичній машині, що й сервер, проте, залежно від архітектурних потреб, можливі й інші варіанти розгортання.

— Zabbix-агент. Це спеціалізоване програмне забезпечення, що встановлюється на стороні клієнта для забезпечення комунікації з сервером. Агент здійснює активний моніторинг локальних ресурсів та системних подій на хості, передаючи відповідні дані на сервер. Важливо зазначити, що збір даних відбувається вибірково — агент відстежує лише ті параметри, які були попередньо сконфігуровані адміністратором[13].

Завдяки такій архітектурі Zabbix здатен у режимі реального часу контролювати тисячі параметрів фізичних серверів, віртуальних машин, програмних додатків та мережевого обладнання. Це дозволяє реалізувати проактивний підхід до обслуговування, виявляючи та усуваючи несправності ще до того, як вони негативно вплинуть на роботу користувачів.

2.3 Механізми збору та захисту даних

Zabbix-агент виконує роль первинного збирача метрик, акумулюючи інформацію про стан обладнання (наприклад, завантаження ЦП) та роботу служб (HTTP, FTP, SSH). Зібрані дані передаються на головний сервер, де вони структуруються для подальшої візуалізації у вигляді таблиць або графіків та зберігаються у реляційних базах даних .

Архітектура Zabbix передбачає два основні методи обміну даними між сервером та агентом:

1. Пасивний режим (Polling/Опитування): Ініціатором виступає сервер (або проксі), який надсилає запит агенту на отримання конкретного значення. Агент обробляє запит і повертає результат .

2. Активний режим (Trapping/Перехоплення): Агент самостійно запитує у сервера список актуальних перевірок, після чого періодично надсилає зібрані дані згідно із заданим розкладом .

Система оснащена гнучким механізмом інформування про інциденти. Хоча найбільш поширеним каналом залишається електронна пошта, Zabbix дозволяє інтегрувати сповіщення через SMS, телефонні дзвінки, месенджери (наприклад, Telegram) та інші сервіси . Формат повідомлень піддається кастомізації, що дозволяє передавати адміністраторам лише релевантну інформацію про подію[14].

Починаючи з версії 3.0, Zabbix підтримує захист трафіку за стандартом TLS (версії 1.2 або 1.3, залежно від бібліотеки), що гарантує безпеку даних при передачі. Підтримуються два методи аутентифікації:

— На основі попереднього спільного ключа (PSK).

— На основі сертифікатів .

Шифрування можна налаштувати для всіх компонентів інфраструктури: серверів, проксі, агентів, утиліт (sender/get) та взаємодії з базою даних. Важливою архітектурною перевагою є використання єдиного порту для обробки як зашифрованого, так і відкритого трафіку, що спрощує налаштування міжмережевих екранів (firewall) . Система дозволяє гібридні конфігурації, де частина агентів працює через PSK, частина через сертифікати, а інші — без шифрування.

Обмеження та вплив на продуктивність: Попри переваги, використання шифрування має низку технічних нюансів: Зберігання ключів: Приватні та спільні ключі зберігаються у БД та зчитуються компонентами у вигляді звичайного тексту, що вимагає додаткових заходів безпеки на рівні доступу до сервера .Мережеве виявлення: Процес Network Discovery не підтримує шифрування. Якщо агент налаштований на прийом виключно захищених

з'єднань, виявлення не спрацює. Продуктивність: Оскільки кешування сеансів TLS наразі не реалізовано, кожне з'єднання вимагає повного «рукостискання» (handshake). Це суттєво збільшує затримки. Наприклад, при мережевій затримці (latency) пакету в 100 мс, стандартне TCP-з'єднання триватиме близько 200 мс, тоді як з використанням TLS цей час може зрости до 1000 мс і більше.

2.4 Розширені можливості: Бази даних та Веб-хуки

Функціонування будь-якого сервера, фізичного чи віртуального, спрямоване на забезпечення роботи програмного забезпечення. Оскільки інстальовані сервіси безпосередньо впливають на стабільність усієї системи, комплексний моніторинг не повинен обмежуватися лише апаратними показниками (CPU, RAM, HDD), а має охоплювати й програмний рівень[15].

Zabbix надає можливості для спостереження за більшістю популярного ПЗ, однак це вимагає попереднього налаштування. Для моніторингу баз даних (БД) часто необхідно встановити додатковий компонент — ODBC (Open Database Connectivity). Це стандарт відкритого підключення, який підтримується такими системами як Oracle, PostgreSQL, MySQL, MSSQL, Sybase ASE, SAP HANA та DB2 .

Специфіка роботи з ODBC:

— Сумісність драйверів: Кожна СКБД має унікальні конектори, функціонал яких може суттєво відрізнятися і не завжди повністю відповідати стандарту ODBC. Тому налаштування, актуальні для однієї бази, можуть не спрацювати для іншої . Важливо також перевіряти сумісність версії драйвера з поточною версією Zabbix-сервера.

— Архітектура взаємодії: Zabbix не встановлює тісного зв'язку з базою даних. Він генерує запит, який передається на ODBC, а той, у свою чергу, звертається до БД через мережевий API. Тайм-аут у налаштуваннях зазвичай

стосується часу очікування входу на ODBC, а не виконання самого запиту . Схематично цей процес зображено на рис. 2.1.

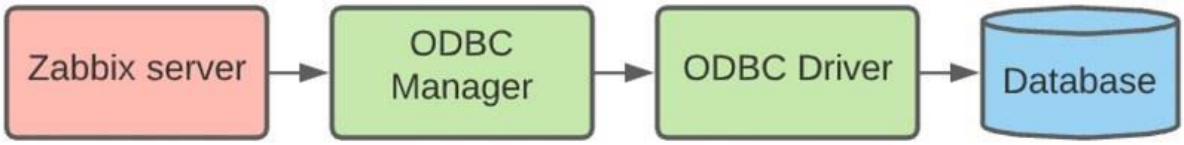


Рисунок 2.2 – Схема маршрутизації запиту від системи Zabbix до бази даних через інтерфейс ODBC

— Налаштування та перевірка: Після інсталяції драйвера необхідно налаштувати конфігураційні файли odbc.ini та odbcinst.ini, а також коректно вказати ім'я джерела даних (DSN). Помилка в DSN унеможливить підключення . Діагностику з'єднання можна провести безпосередньо з консолі сервера за допомогою утиліти isql[16]. Наприклад, для MSSQL перевіряється доступність системної бази msdb, рис. 2.3. Успішне підключення свідчить про коректність налаштувань драйвера та прав доступу .

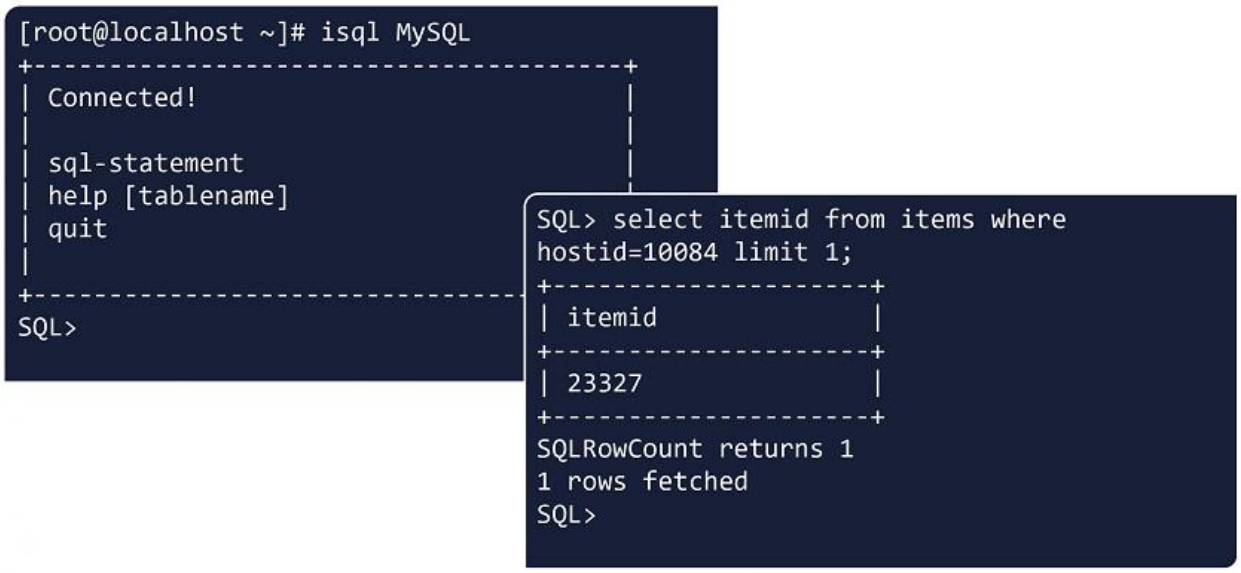


Рисунок 2.3 – Верифікація налаштувань ODBC-драйвера шляхом виконання тестового підключення утилітою isql

Веб-хуки — це механізм автоматичної реакції на події через HTTP/HTTPS виклики, що дозволяє змінювати поведінку веб-додатків або інтегрувати їх між собою. У контексті Zabbix це означає, що при виникненні специфічної події (наприклад, збою) система може автоматично надіслати запит до стороннього сервісу через його API .

Веб-хуки в Zabbix реалізовані на мові JavaScript. Це знижує поріг входження, оскільки не вимагає вивчення внутрішнього синтаксису Zabbix, а готові скрипти легко знайти у відкритому доступі.

Скрипт виконує послідовність дій для досягнення результату. Наприклад, для створення тікета в Service Desk система спочатку авторизується для отримання токена, потім створює запит на тікет і, нарешті, додає коментар з деталями інциденту. Схему передачі даних відображено на рис. 2.4.

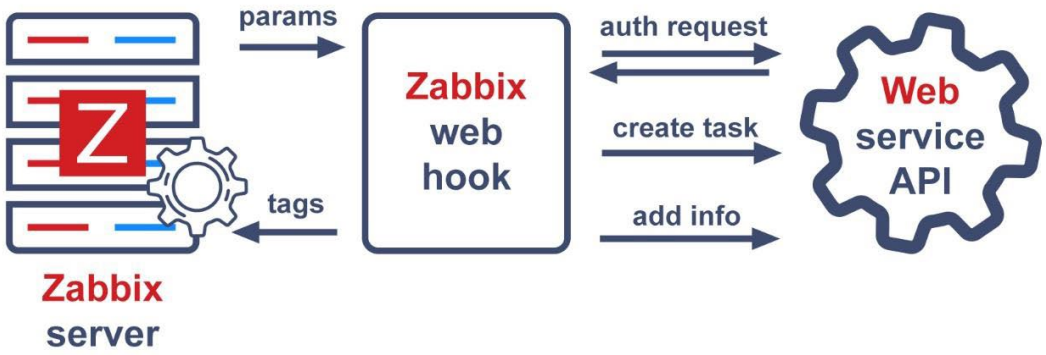


Рисунок 2.4 – Алгоритм інформаційної взаємодії між системою Zabbix, модулем веб-хука та зовнішнім API інтерфейсом

Zabbix постачається з набором готових веб-хуків для популярних сервісів. Їх налаштування є тривіальним і зазвичай вимагає лише генерації API-ключа на стороні сервісу та внесення URL-адреси кінцевої точки у параметри Zabbix . Створення власних інтеграцій вимагає глибшого розуміння структури API конкретного сервісу та принципів взаємодії Zabbix із зовнішніми системами, оскільки методи запитів можуть відрізнятися .

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОНІТОРИНГУ НА БАЗІ ZABBIX

3.1 Підготовка серверного середовища та розгортання репозиторіїв

Для успішного розгортання системи Zabbix необхідно забезпечити наявність операційної системи сімейства Linux. Офіційна документація декларує підтримку широкого спектра дистрибутивів, серед яких: AlmaLinux, CentOS, Debian, Oracle Linux, Raspberry Pi OS, Red Hat Enterprise Linux, Rocky Linux, SUSE Linux Enterprise Server та Ubuntu (включно з версіями для архітектури arm64).

Для виконання роботи з налаштування моніторингу було сформовано наступний стек програмного забезпечення:

— Версія Zabbix-сервера 7.0 LTS. Вибір зумовлений тим, що це найактуальніша версія з довгостроковою підтримкою (Long Term Support). На відміну від проміжних релізів, версія 7.0 гарантує стабільність та отримання оновлень безпеки протягом 5 років, а також містить останні архітектурні вдосконалення порівняно з попередніми генераціями (6.0 та 6.4).

— Операційна система Ubuntu 24.04 LTS (Noble Numbat). Це найновіший реліз із довгостроковою підтримкою від Canonical. Вибір даної ОС базується на її домінуючій позиції на ринку Linux-систем. Висока популярність Ubuntu гарантує наявність великої бази знань спільноти, що суттєво спрощує пошук рішень для усунення можливих технічних проблем чи виконання нестандартних налаштувань.

— Система керування базами даних MySQL. Було обрано одну з найбільш розповсюджених реляційних баз даних. Її перевагами є простота в адмініструванні, вичерпна документація та перевірена часом стабільність, що робить її оптимальним сховищем для даних моніторингу Zabbix.

— Веб-сервер Nginx. Вибір на користь Nginx замість Apache2 зроблено через його вищу продуктивність при обробці статичного контенту, гнучкість конфігурації та простоту налаштування правил переадресації (reverse proxy), що є важливим для коректної роботи веб-інтерфейсу Zabbix.

Для розгортання програмного комплексу було підготовлено віртуальний сервер з хост-іменем «test-zabbix». Перед початком інсталяції пакетів критично важливо актуалізувати стан системних репозиторіїв та оновити наявні компоненти ОС.

Для цього в терміналі виконуються наступні команди, рис.3.1.

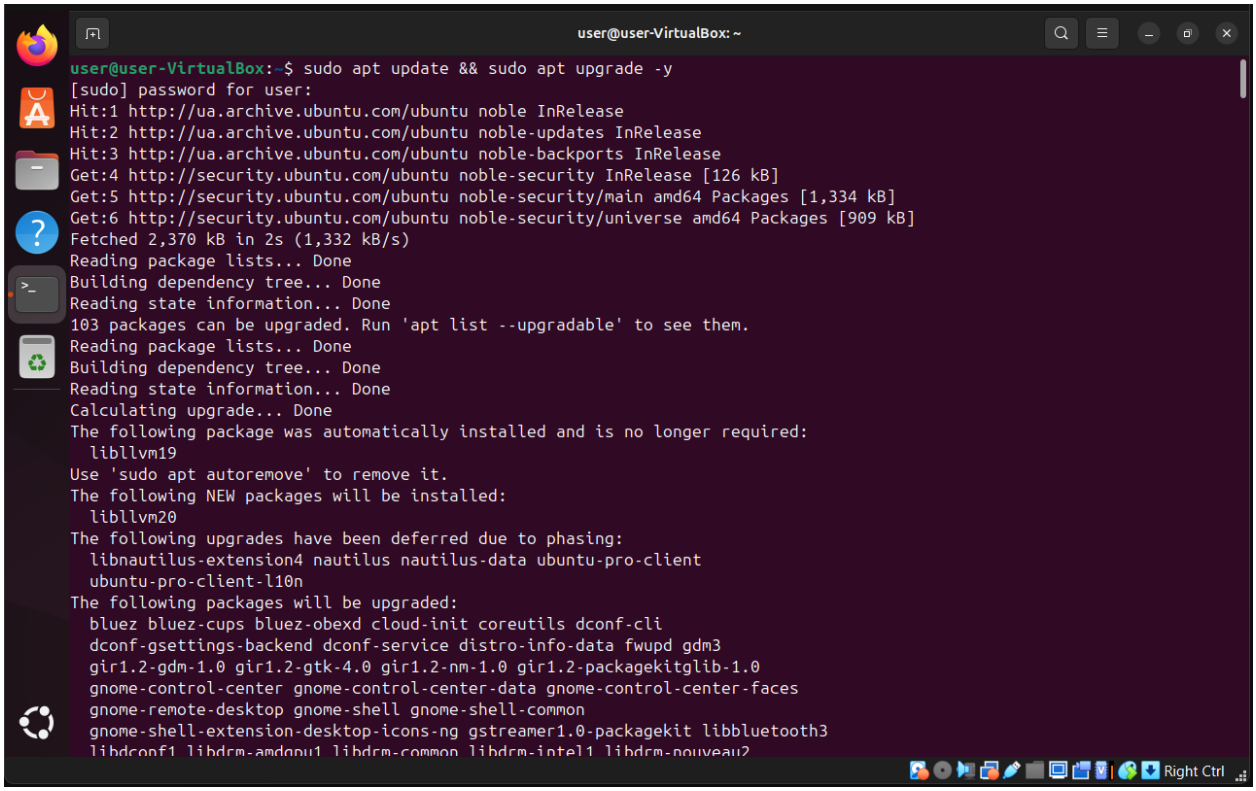


Рисунок 3.1 – Команди для оновлення компонентів ОС.

Наступним кроком ми завантажуємо офіційний пакет Zabbix 7.0 LTS, рис 3.2.

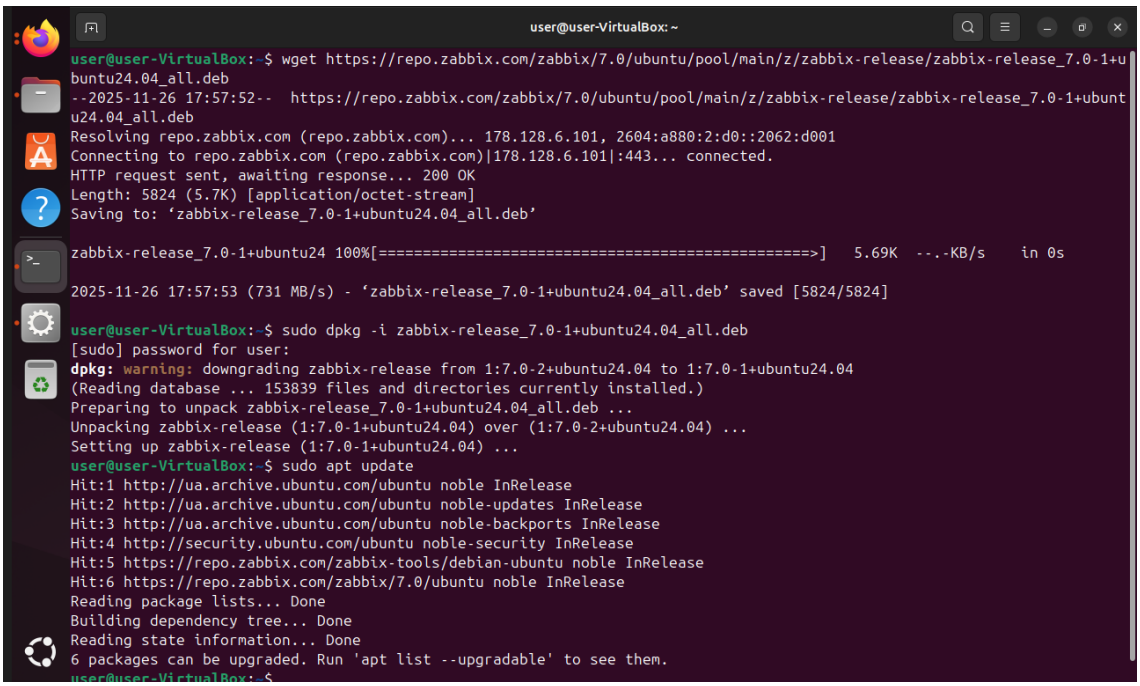


Рисунок 3.2 – Команди для завантаження та встановлення пакету Zabbix

Далі ініціюємо інсталяцію LEMP-стеку (Linux, Nginx, MySQL, PHP), адаптованого для потреб Zabbix. Нижче наведено класифікацію пакетів за їх функц Ці компоненти відповідають за логіку обробки даних, збереження конфігурації, історію метрик та забезпечують графічний інтерфейс користувача (GUI) та взаємодію через веб-протоколи, рис.3.3.

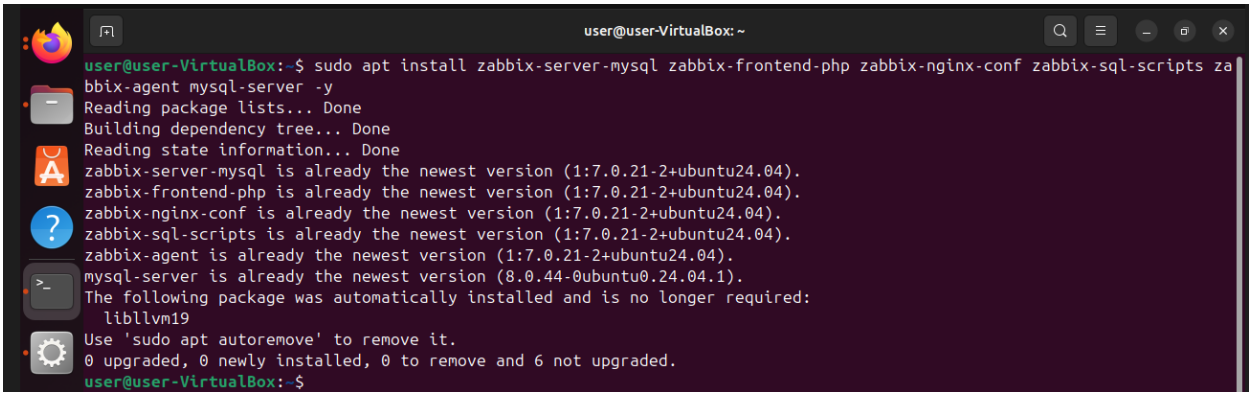


Рисунок 3.3 – Команди для інсталяції LEMP-стеку

— zabbix-server-mysql: Центральний демон (daemon) системи моніторингу. Він виконує функції збору даних (polling/trapping), обробки отриманих значень, обчислення тригерів та ініціювання сповіщень. Суфікс -

mysql вказує на те, що бінарний файл скомпільовано з підтримкою бібліотек для взаємодії саме з СУБД MySQL/MariaDB як сховищем даних.

— mysql-server: Сервер системи управління базами даних (СУБД). Виконує роль персистентного сховища для Zabbix Server. У ньому зберігається вся конфігурація (хости, шаблони, користувачі) та зібрані історичні дані.

— zabbix-sql-scripts: Пакет, що містить DDL (Data Definition Language) та DML (Data Manipulation Language) скрипти. Вони необхідні для ініціалізації схеми бази даних — створення таблиць, індексів та наповнення початковими даними, необхідними для запуску сервера.іональним призначенням в архітектурі системи.

— zabbix-frontend-php: Веб-інтерфейс системи, написаний на мові PHP. Забезпечує візуалізацію даних (графіки, дашборди), адміністрування системи та налаштування правил моніторингу. Виступає клієнтом до бази даних та Zabbix Server API.

— zabbix-nginx-conf: Конфігураційний файл інтеграції для веб-сервера Nginx. Він містить директиви server або location, необхідні для коректної обробки PHP-запитів фронтенду Zabbix. Наявність цього пакету передбачає використання Nginx замість Apache як веб-сервера.

3.2 Розгортання бази даних

На першому етапі було виконано підготовку середовища: оновлення системних пакетів, додавання офіційного репозиторію Zabbix та встановлення LAMP-стеку. Використання офіційного репозиторію гарантує відсутність конфліктів залежностей.

Критично важливим компонентом є база даних. Процес імплементації рівня зберігання даних Zabbix є двостадійним алгоритмом, що розпочинається з конфігурації середовища СУБД, яка включає створення ізольованої бази

даних з бінарною коляцією utf8mb4_bin та делегування повноважень виділеному користувачеві з тимчасовим дозволом на створення збережених процедур - log_bin_trust_function_creators, рис.3.3 - 3.4.

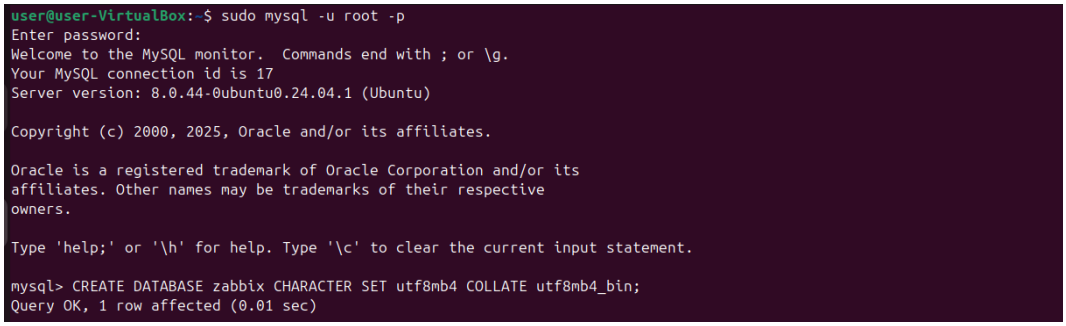


Рисунок 3.3 – Створення бази даних

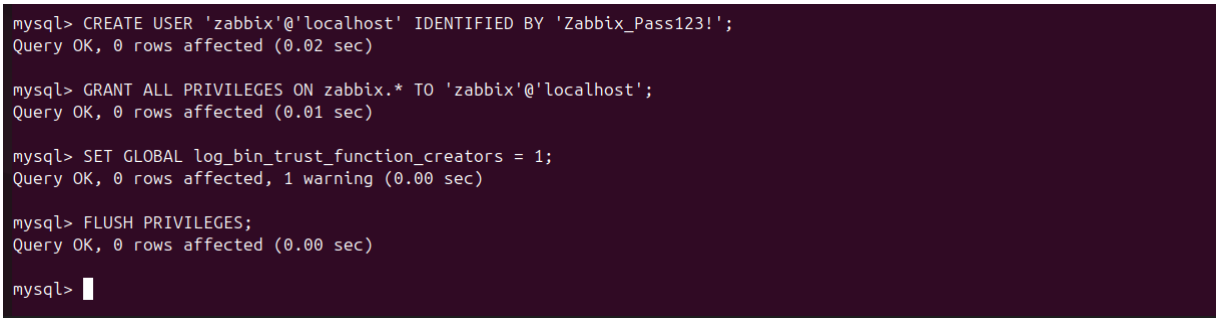


Рисунок 3.4 – Налаштування прав доступу в середовищі MySQL

Завершальним етапом є розгортання реляційної схеми, рис.3.5, під час якого попередньо підготовлений SQL-дамп декомпресується та імпортується в ініціалізоване середовище, формуючи необхідну для функціонування системи моніторингу архітектуру таблиць та тригерів.

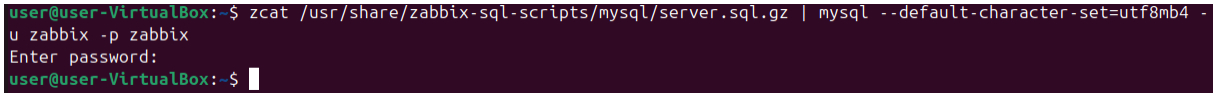


Рисунок 3.5 – Розгортання реляційної схеми в середовищі MySQL

Після завершення імпорту, гарною практикою вважається повернення налаштування set global log_bin_trust_function_creators = 0, в консолі MySQL для відновлення стандартного рівня безпеки, рис.3.6.

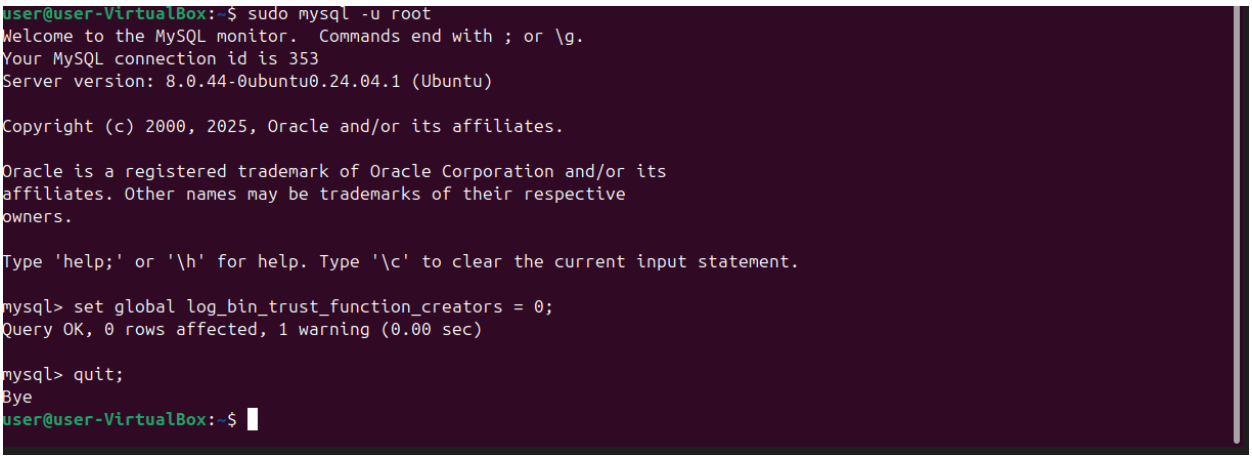


Рисунок 3.6 – Відновлення стандартного рівня безпеки в середовищі MySQL

Цей крок реалізує імплементацію облікових даних у конфігурацію прикладного рівня, рис.3.7. Редагування файлу zabbix_server.conf встановлює необхідний параметр автентифікації, що дозволяє бекенд-процесу Zabbix ініціювати успішне з'єднання з системою управління базами даних (СУБД). Активуючи директиву DBPassword, адміністратор забезпечує відповідність параметрів доступу тим, що були визначені на етапі створення привілейованого користувача бази даних (GRANT ALL PRIVILEGES), тим самим замикаючи контур взаємодії між сервером моніторингу та його підсистемою зберігання даних.

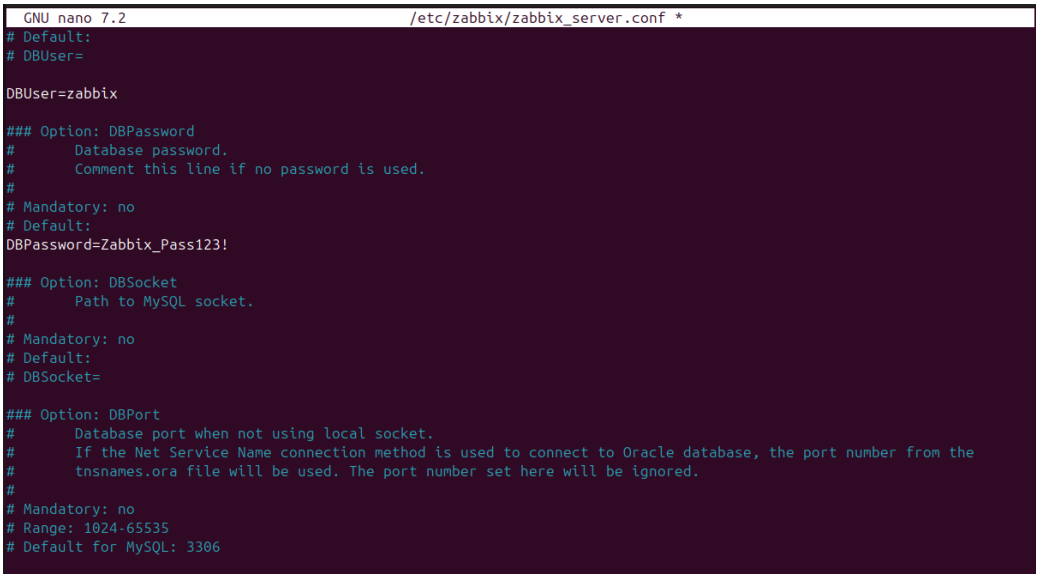


Рисунок 3.7 – Імплементація облікових даних у конфігурацію прикладного рівня

3.3 Налаштування Firewall, Nginx та оркестрація служб.

Завершальна стадія розгортання охоплює конфігурацію мережевого периметра та ініціалізацію прикладних сервісів. Налаштування міжмережевого екрана (UFW) реалізує політику вибіркової доступності, експонуючи порти для HTTP-трафіку (80) та протоколу обміну даними Zabbix (10050/10051), зберігаючи при цьому канал адміністративного доступу (SSH). Паралельно здійснюється параметризація веб-сервера Nginx шляхом визначення директив прослуховування сокетів та віртуального хостингу. Фінальна команда здійснює перезавантаження демонів для імплементації змін у конфігураційних файлах та реєструє служби в автозавантаженні (systemd enable), забезпечуючи персистентність функціонування системи моніторингу після перезавантаження операційної системи.

Налаштування Uncomplicated Firewall — стандартний інструмент керування netfilter в Ubuntu, рис.3.8.

```
user@user-VirtualBox:~$ sudo ufw allow 80/tcp
Rules updated
Rules updated (v6)
user@user-VirtualBox:~$ sudo ufw allow 10050/tcp
Rules updated
Rules updated (v6)
user@user-VirtualBox:~$ sudo ufw allow 10051/tcp
Rules updated
Rules updated (v6)
user@user-VirtualBox:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
user@user-VirtualBox:~$ sudo ufw enable
Firewall is active and enabled on system startup
user@user-VirtualBox:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
10050/tcp                  ALLOW       Anywhere
10051/tcp                  ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
10050/tcp (v6)             ALLOW       Anywhere (v6)
10051/tcp (v6)             ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Рисунок 3.8 – Налаштування Uncomplicated Firewall

Мета — дозволити трафік лише на критично важливі порти, блокуючи все інше (принцип мінімальних привілеїв).

— 80/tcp (HTTP): Відкриває доступ для веб-браузера, щоб ви могли зайти в веб-інтерфейс Zabbix.

— 10050/tcp (Zabbix Agent): Порт, який слухає Агент. Сервер Zabbix підключається сюди, щоб запитати дані (Passive Check).

— 10051/tcp (Zabbix Server/Trapper): Порт, який слухає Сервер. Агенти підключаються сюди, щоб надіслати дані (Active Check) або повідомлення про помилки.

— ssh (22/tcp): Критично важливо для віддаленого керування сервером. Без цього ви втратите доступ до консолі після команди enable.

Налаштування файлу Zabbix, рис.3.9, який спеціально створено для інтеграції з Nginx, що переводить конфігурацію зі стану "шаблону" в стан "активної директиви".



Рисунок 3.9 – Конфігурація файлу Zabbix для інтеграції з Nginx

— listen 80;: Вказує Nginx "слухати" вхідні з'єднання на порту 80 (стандартний веб-порт) по протоколу IPv4.

— server_name example.com;: Визначає віртуальний хост. Це вказує Nginx, що цей блок налаштувань застосовується, коли до сервера звертаються за доменним ім'ям.

Використання системного менеджера systemd, рис.3.10, для керування життєвим циклом daemon.



Рисунок 3.10 – Команди системного менеджера systemd

— restart: Зупиняє та знову запускає процеси. Це необхідно, щоб вони перечитали змінені конфігураційні файли (zabbix_server.conf, nginx.conf, php.ini). Без перезапуску зміни пароля БД чи портів не наберуть чинності.

— enable: Створює символічні посилання (symlinks) у цільових директоріях systemd (наприклад, multi-user.target.wants). Це гарантує, що після фізичного перезавантаження сервера (reboot) всі компоненти Zabbix запустяться автоматично без вашого втручання.

— php8.3-fpm: Менеджер процесів PHP (FastCGI Process Manager). Саме він виконує PHP-код інтерфейсу Zabbix і передає результат Nginx-у.

Останній крок це команда systemctl status, рис.3.11, яка ініціює запит до підсистеми ініціалізації systemd для отримання поточного експлуатаційного стану юніта zabbix-server. Ця процедура виконує функцію діагностичного аудиту, дозволяючи адміністратору верифікувати успішність запуску демона (Active state), проаналізувати ієрархію процесів (PID) та переглянути останні записи в системному журналі (stdout/stderr log tail) для виявлення потенційних конфліктів конфігурації або помилок підключення до суміжних сервісів (СУБД) у реальному часі.

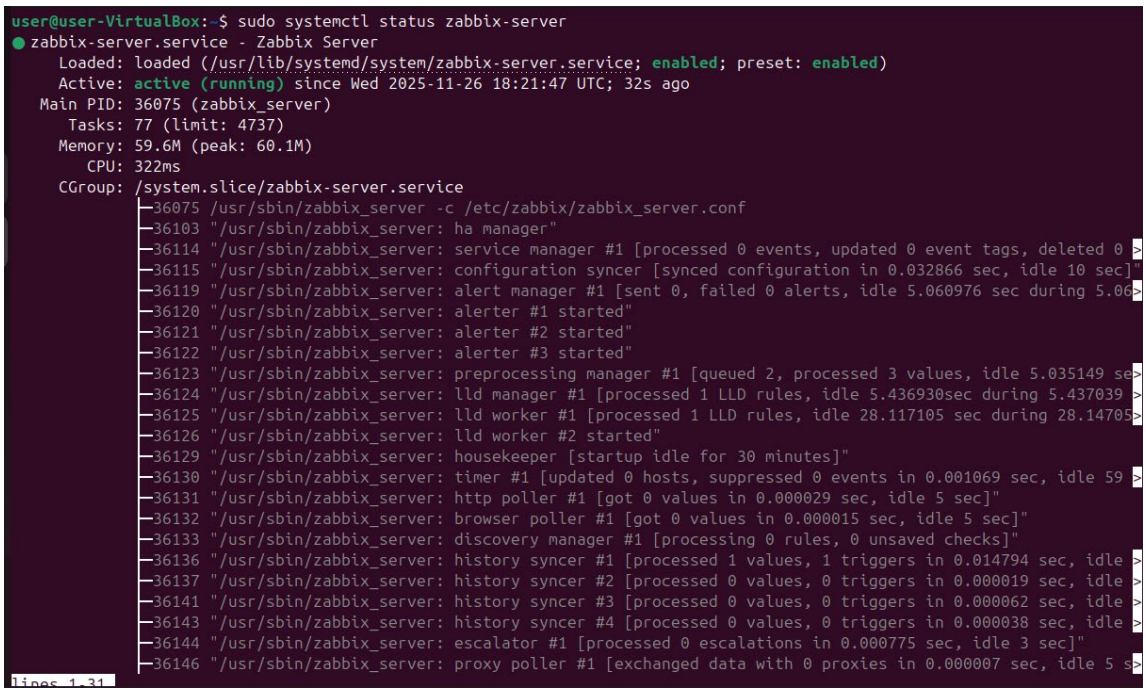


Рисунок 3.11 – Перевірка статусу Zabbix сервера

3.4 Ініціалізація веб-інтерфейсу та фінальна конфігурація Zabbix Frontend

Після успішного розгортання серверної частини (Backend), підготовки схеми бази даних та конфігурації веб-сервера Nginx на попередніх етапах, необхідним кроком є інтеграція рівня представлення (Frontend). Цей процес реалізується через веб-майстер встановлення, метою якого є перевірка сумісності середовища та генерація фінального конфігураційного файлу PHP (zabbix.conf.php). Цей файл забезпечує програмний зв'язок між візуальною оболонкою, сервером моніторингу та системою управління базами даних.

Нижче наведено покроковий опис процесу конфігурації:

1. Вхідна точка процедури встановлення. Система ідентифікує версію програмного забезпечення та пропонує вибір мови локалізації інтерфейсу. Цей етап підтверджує, що веб-сервер Nginx коректно обробляє PHP-скрипти Zabbix, рис.3.12.

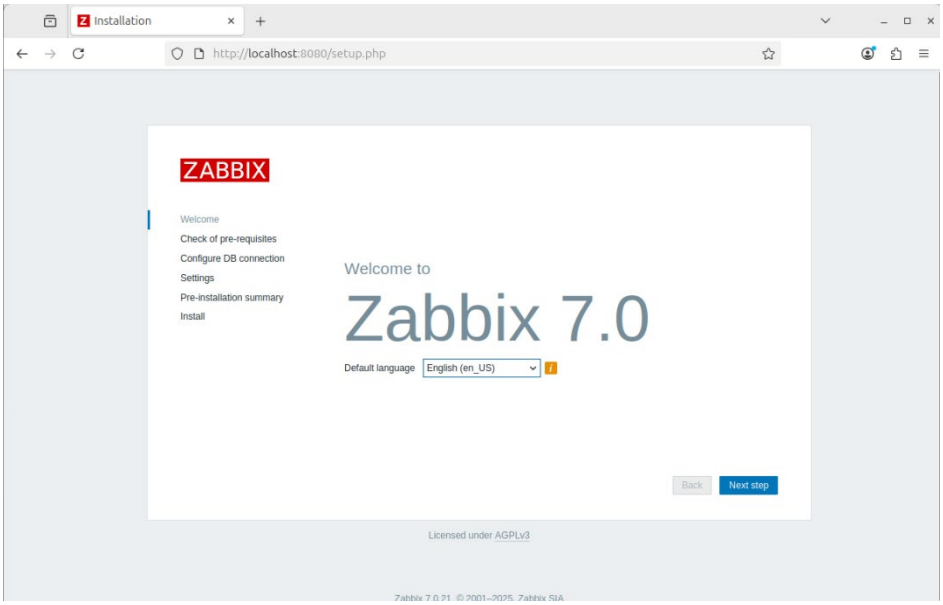


Рисунок 3.12 – Ініціалізація майстра налаштування Zabbix 7.0

2. Автоматизований аудит конфігурації середовища виконання PHP. Система здійснює перевірку критичних параметрів (ліміти пам'яті memory_limit, час виконання скриптів max_execution_time) та наявність необхідних бібліотек (bcmath, mbstring, підтримка MySQL). Статус "OK" навпроти кожного пункту є обов'язковим для продовження інсталяції та гарантує стабільність роботи інтерфейсу, рис. 3.13.

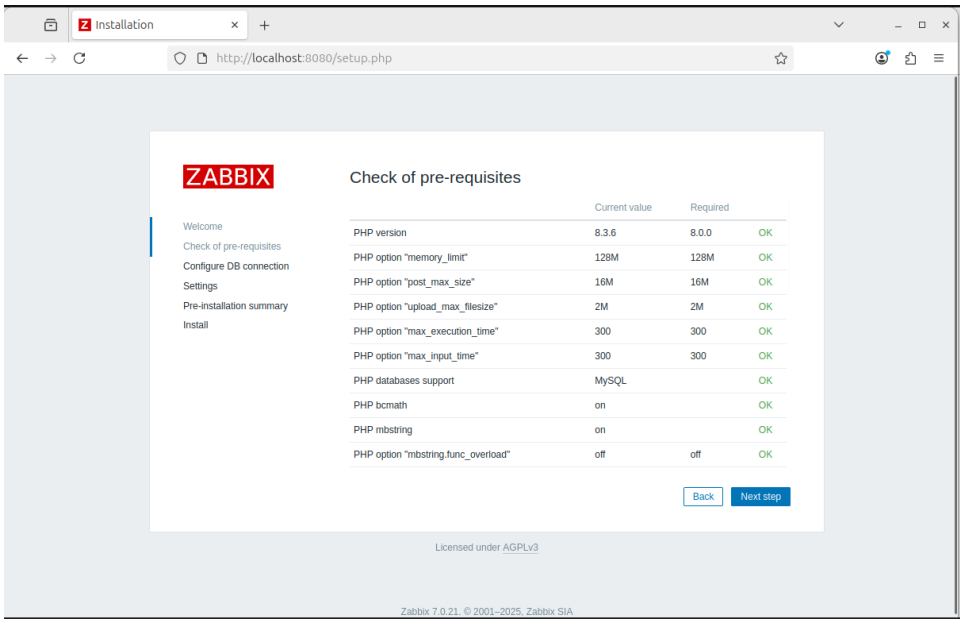


Рисунок 3.13 – Валідація системних вимог (Pre-requisites check)

3. Налаштування параметрів доступу Frontend-частини до СУБД. На цьому етапі вводяться облікові дані (ім'я бази даних zabbix, користувач zabbix та пароль), які були створені раніше через командний рядок MySQL. Це дозволяє веб-інтерфейсу зчитувати зібрані метрики та зберігати зміни в налаштуваннях, рис.3.14.

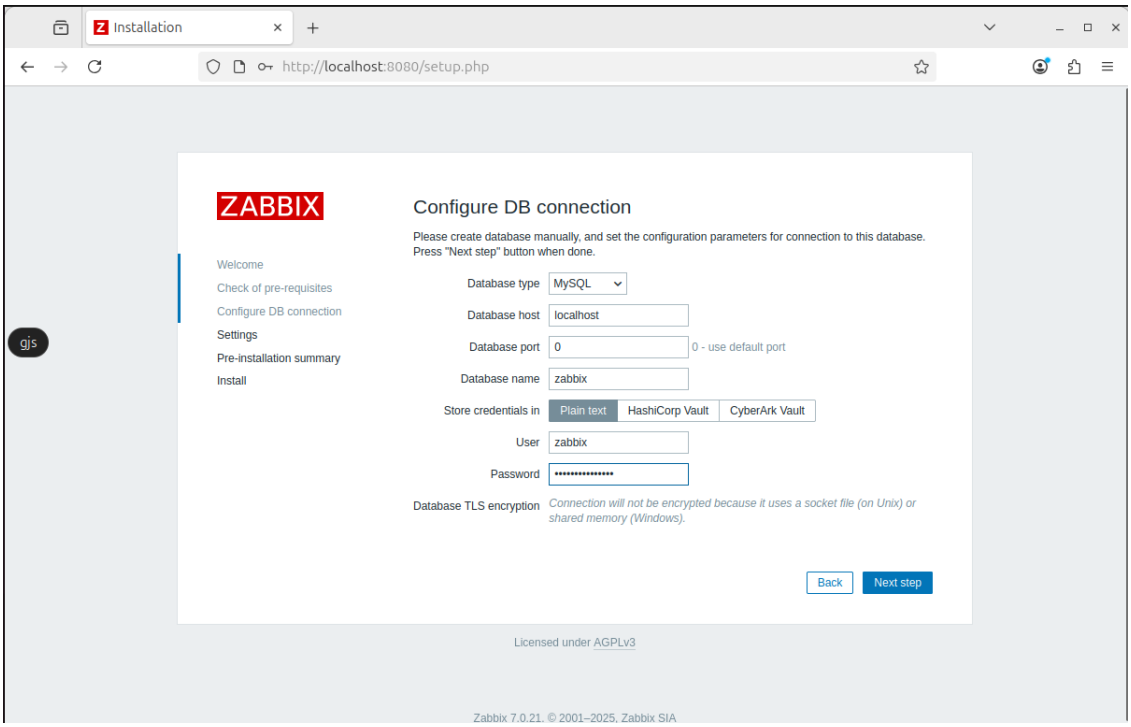


Рисунок 3.14 – Конфігурація підключення до бази даних

4. Визначення ідентифікаторів вузла моніторингу. У полі "Zabbix server name" задається унікальне ім'я інстансу для відображення у звітах. Критично важливим є налаштування часового поясу ("Default time zone"), що забезпечує коректну синхронізацію часових міток подій (timestamps) із реальним часом адміністратора, рис.3.15.

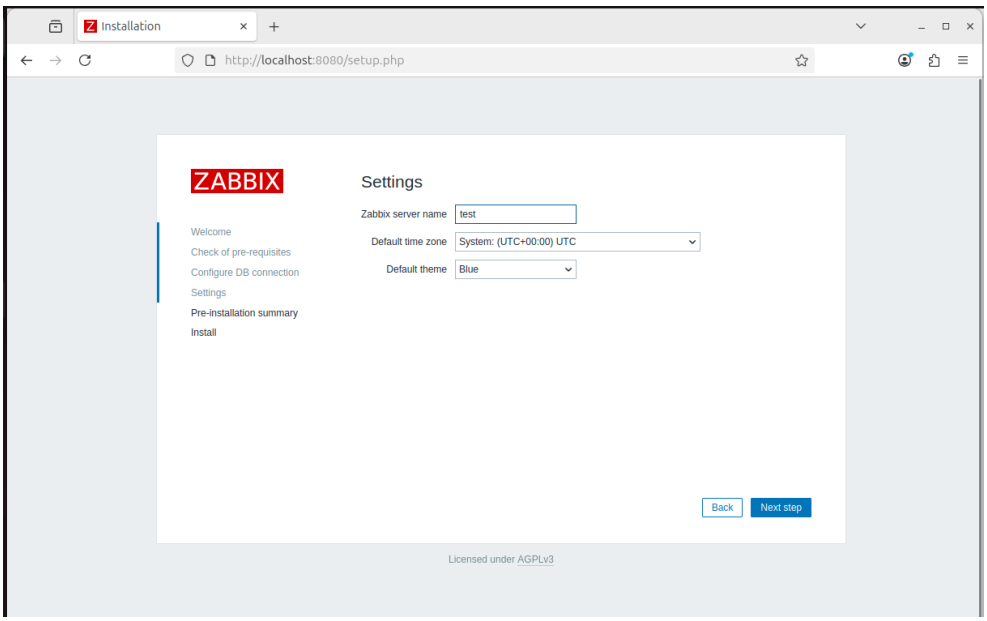


Рисунок 3.15 – Параметризація сервера та локалізація

5. Фіналізація роботи майстра налаштування. Повідомлення про успішне створення файлу conf/zabbix.conf.php свідчить про те, що веб-інтерфейс отримав права на запис у відповідну директорію та зберіг усі введені параметри підключення. Система готова до авторизації користувача, рис.3.16.

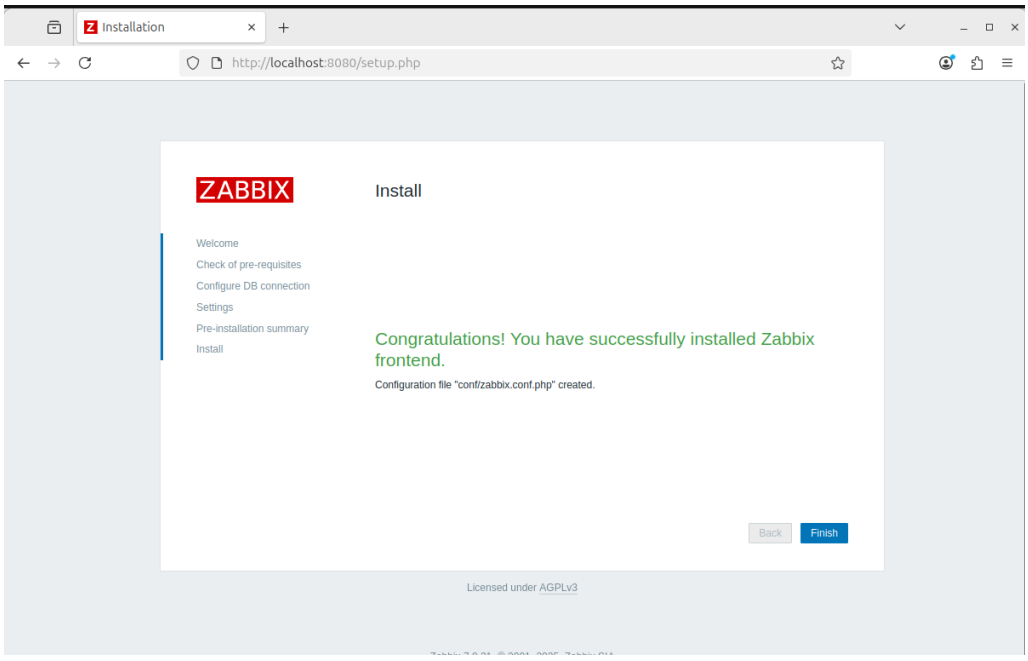


Рисунок 3.16 – Генерація конфігураційного файлу та завершення інсталяції

6. Головна інформаційна панель після входу в систему. Ключовим індикатором успішного розгортання всього комплексу є віджет "System information", де параметр "Zabbix server is running" має значення "Yes". Це підтверджує, що фоновий демон сервера (Port 10051) запущений, активний та успішно комунікує з веб-інтерфейсом і базою даних, рис.3.17.

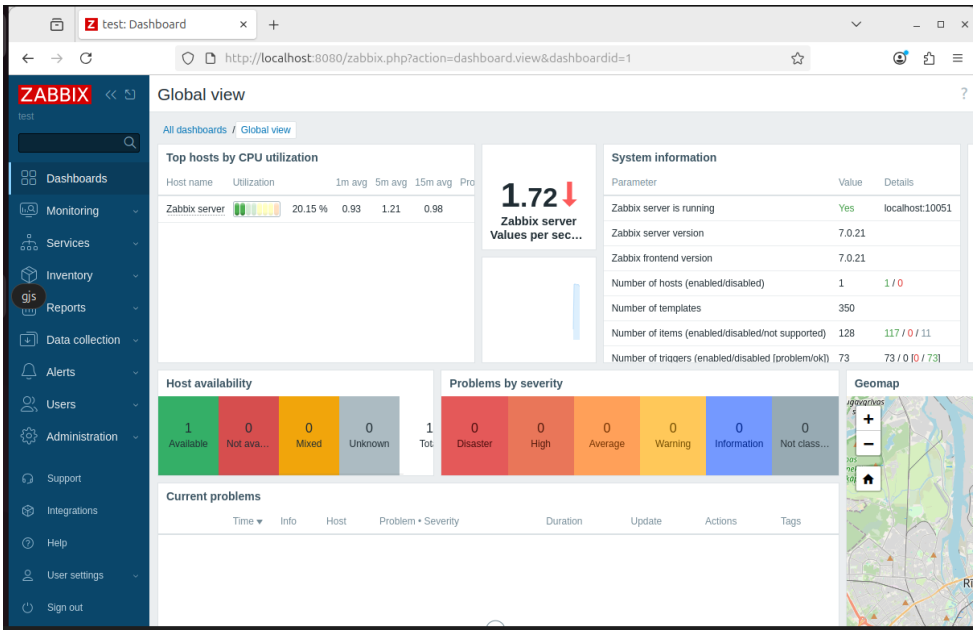


Рисунок 3.17 – Діагностика статусу системи (Dashboard)

3.5 Розгортання агента моніторингу Zabbix Agent 2 та реєстрація цільового хоста

Після успішного запуску серверної частини та веб-інтерфейсу, наступним кроком є організація збору метрик. Для цього використовується архітектура «клієнт-сервер», де на цільовому вузлі встановлюється агент, що передає дані на сервер. У даному розділі розглядається інсталяція сучасної версії агента — Zabbix Agent 2 (написаного на мові Go), його конфігурація на рівні операційної системи та подальша реєстрація в адміністративній панелі Zabbix.

Завантаження та встановлення бінарних файлів агента з офіційного репозиторію за допомогою пакетного менеджера apt, рис.3.18. Використання версії zabbix-agent2 забезпечує розширені можливості моніторингу та кращу продуктивність завдяки багатопоточній архітектурі порівняно з класичним агентом.

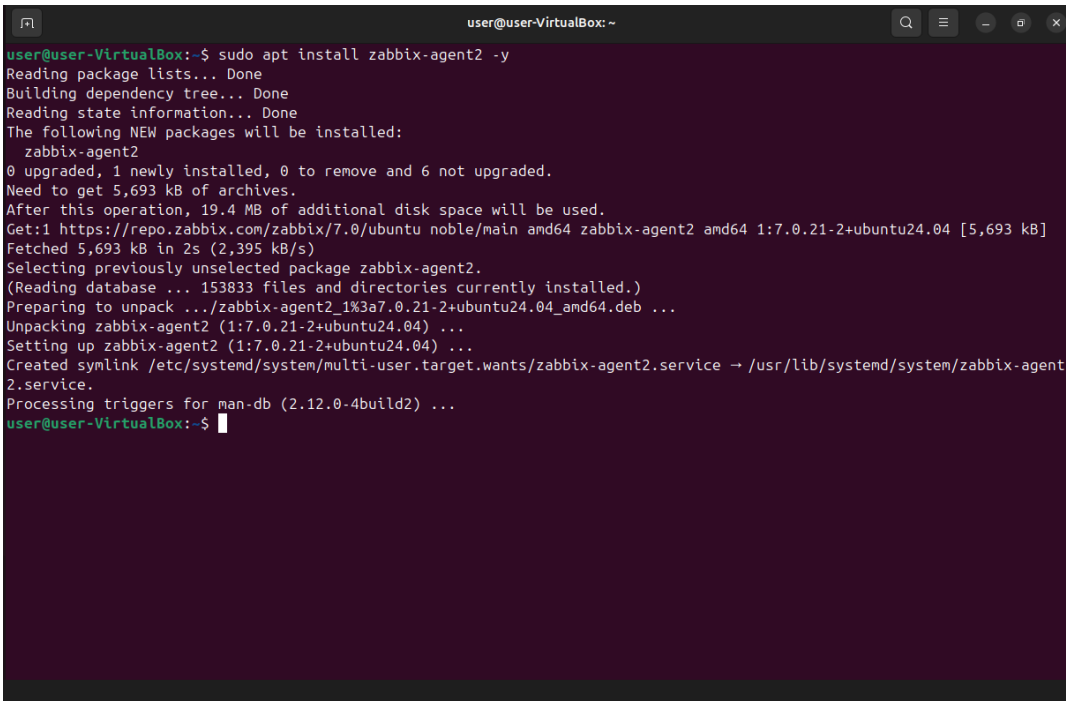


Рисунок 3.18 – Інсталяція пакету Zabbix Agent 2

Редагування конфігураційного файлу /etc/zabbix/zabbix_agent2.conf, рис.3.19. Ключовим етапом є встановлення директиви Hostname=Virtual-Node-01. Це унікальне ім'я слугує ідентифікатором, за яким Zabbix Server розпізнаватиме цей пристрій. Також перевіряється параметр ServerActive, що вказує адресу сервера для активних перевірок.

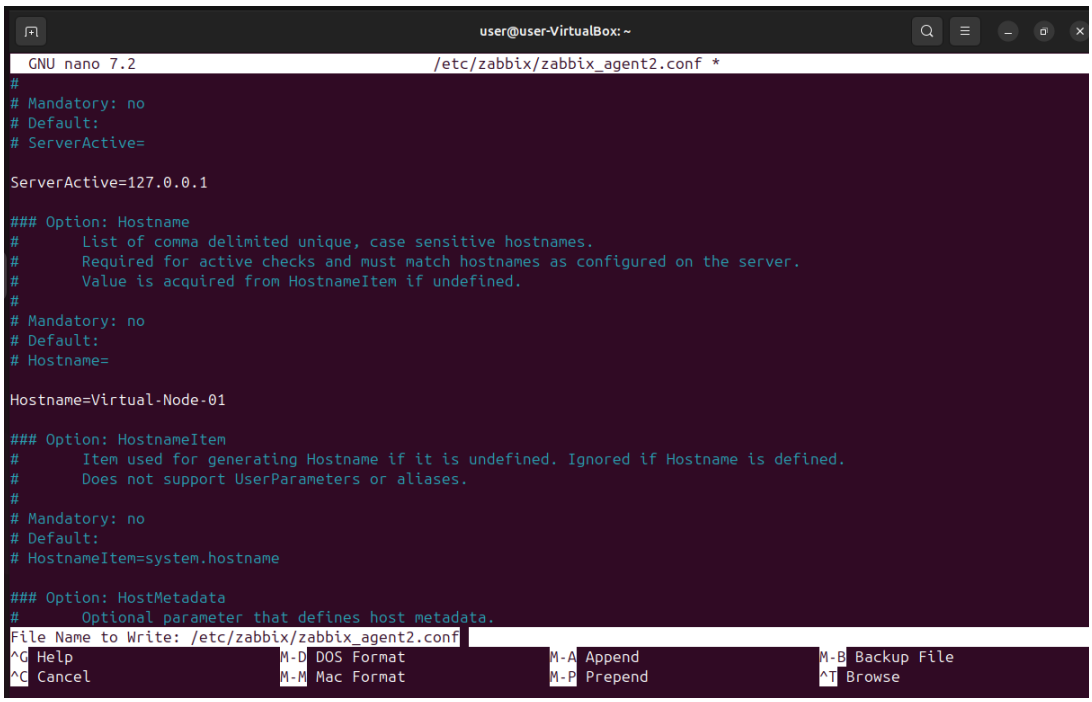


Рисунок 3.19 – Конфігурація параметрів ідентифікації агента

Ініціалізація демона агента засобами systemd, рис.3.20. Команда enable гарантує автоматичний старт служби моніторингу після перезавантаження операційної системи, а restart застосовує зміни, внесені у конфігураційний файл на попередньому етапі.

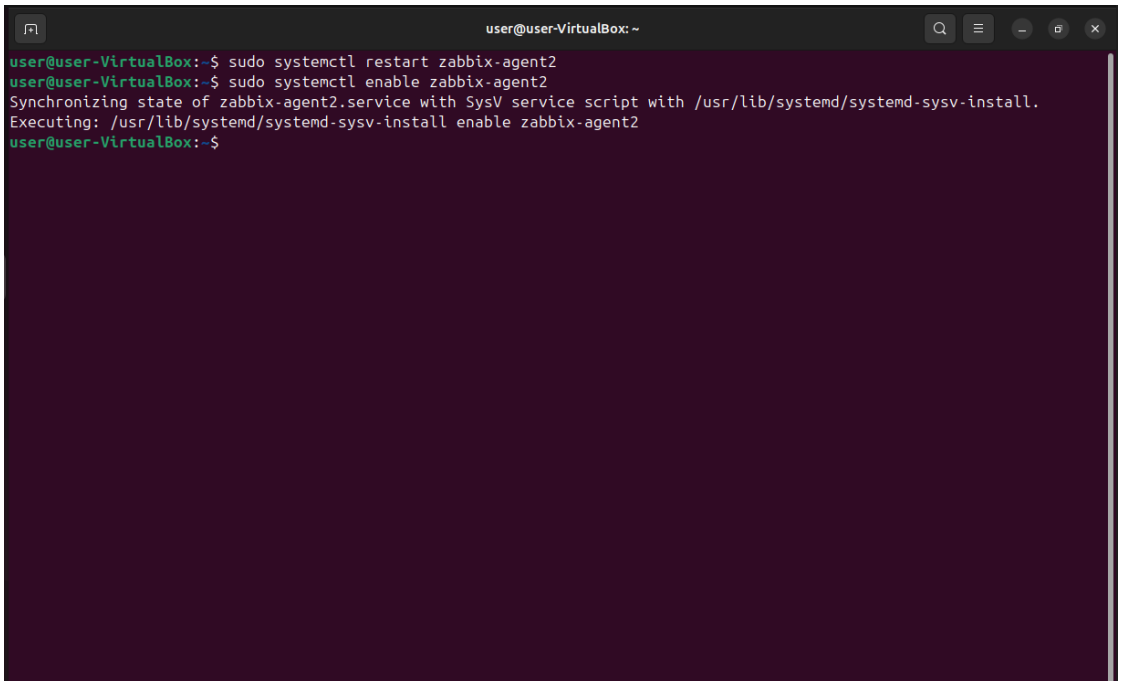


Рисунок 3.20 – Запуск служби та додавання в автозавантаження

Реєстрація нового пристрою в панелі керування Zabbix, рис.3.21. У полі "Host name" вказується значення Virtual-Node-01, яке має строго співпадати з тим, що було прописано у конфігурації агента. Хосту призначається шаблон "Linux by Zabbix agent" для автоматичного збору стандартних метрик та вказується мережевий інтерфейс (127.0.0.1:10050).

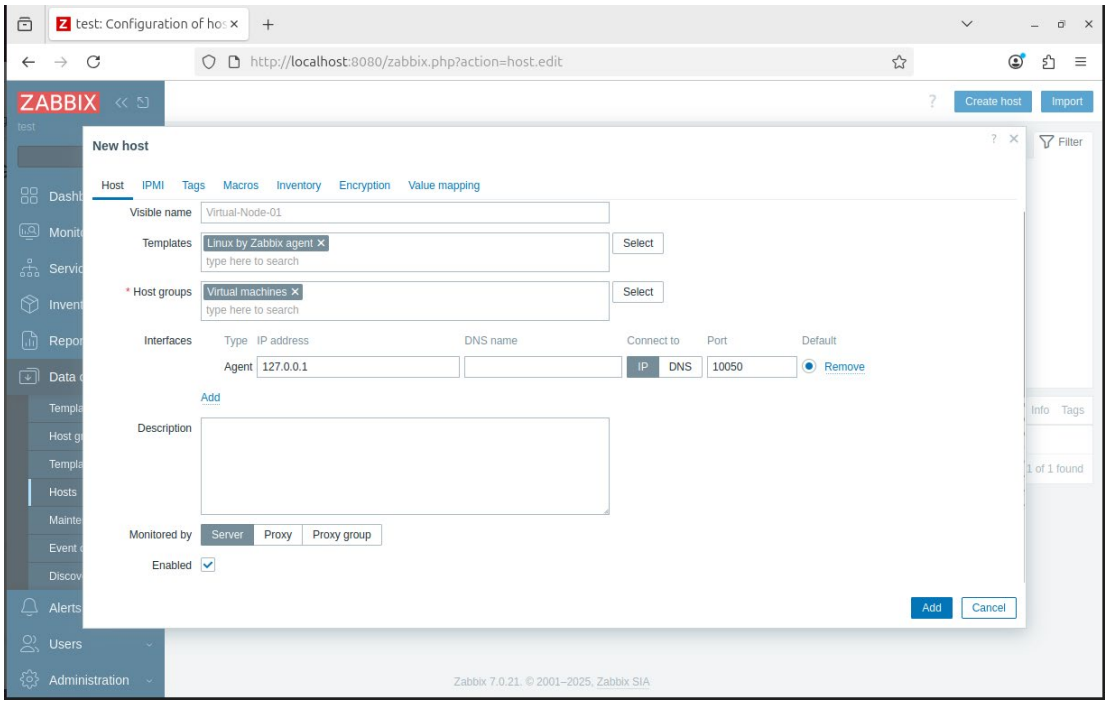


Рисунок 3.21 – Створення об'єкта хоста у веб-інтерфейсі

Відображення нового вузла в списку хостів одразу після створення. На цьому етапі індикатор доступності (Availability ZBX) є сірим, що свідчить про те, що сервер ще не встиг встановити з'єднання з агентом або отримати перші пакети даних, рис.3.22.

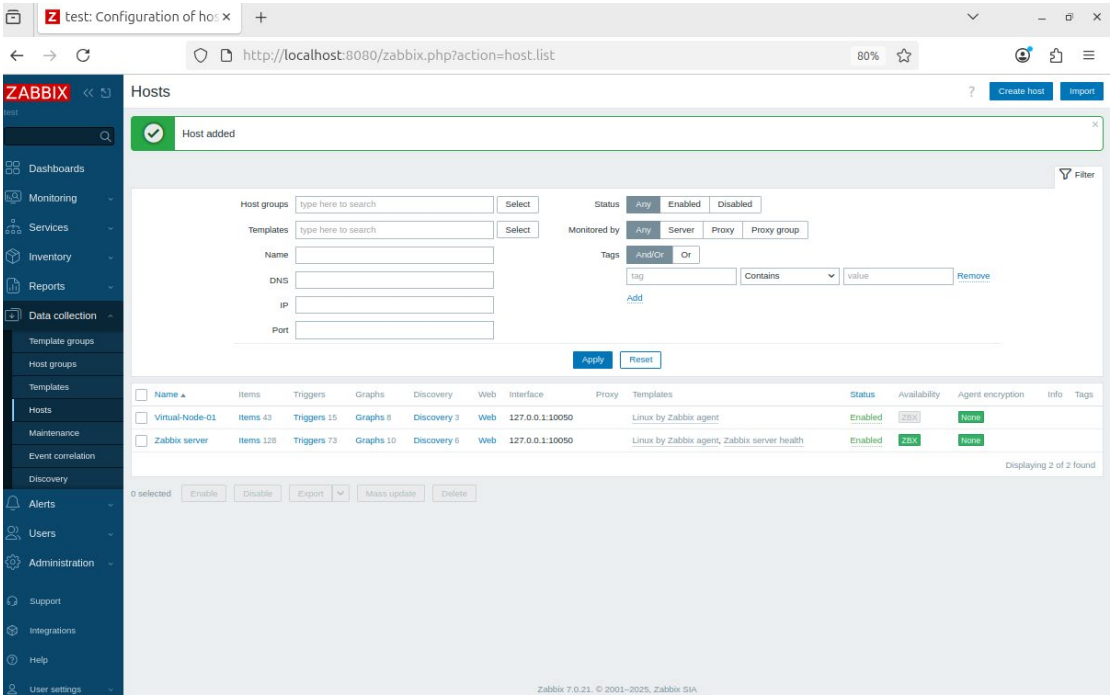


Рисунок 3.22 – Первинний статус хоста після додавання

Підтвердження працездатності системи через деякий час. Індикатор ZBX змінив колір на зелений, що сигналізує про успішний обмін даними між Zabbix Server та Zabbix Agent 2. Це означає, що метрики збираються, тригери обробляються, і хост Virtual-Node-01 знаходиться під повним моніторингом, рис.3.23.

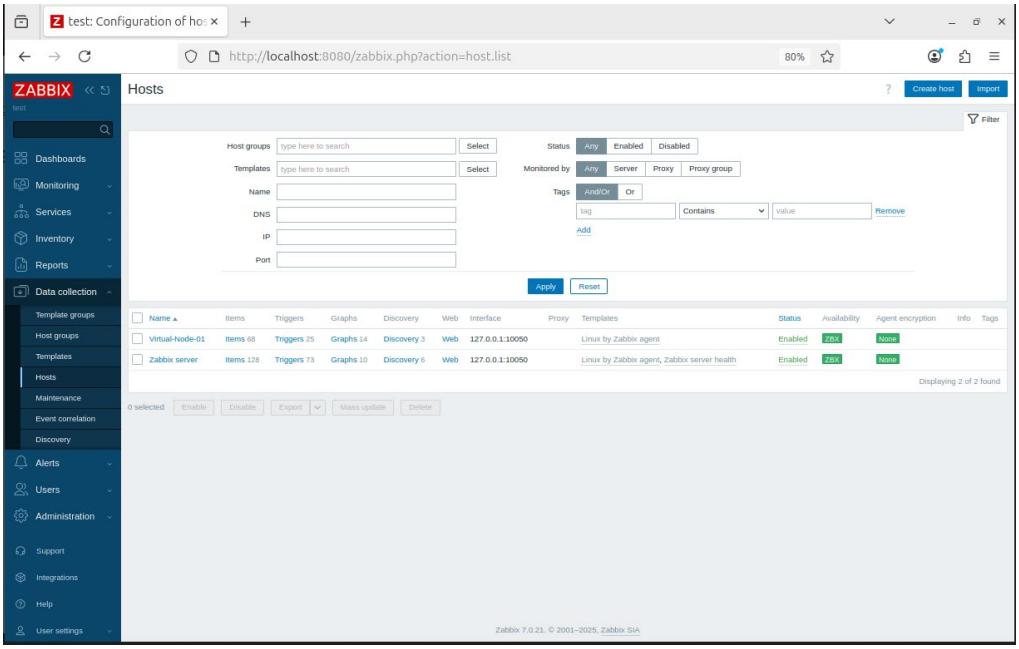


Рисунок 3.23 – Верифікація успішного моніторингу

3.6 Організація прикладного моніторингу СУБД MySQL

Після забезпечення базового моніторингу операційної системи, наступним рівнем є контроль стану прикладного програмного забезпечення (Application Layer). У даному випадку критично важливим є відстеження параметрів роботи системи управління базами даних MySQL, яка виступає сховищем для самого Zabbix. Для реалізації цього завдання необхідно створити окремого сервісного користувача в СУБД з обмеженими правами доступу та налаштувати відповідні макроси авторизації на стороні Zabbix-агента.

Підготовка середовища СУБД через інтерфейс командного рядка. Створюється локальний користувач zbx_monitor та йому надаються специфічні привілеї (REPLICATION CLIENT, PROCESS, SHOW DATABASES, SHOW VIEW), рис.3.24. Такий набір прав є мінімально необхідним для того, щоб плагін моніторингу міг зчитувати статистику продуктивності, не маючи при цьому доступу до зміни самих даних (принцип найменших привілеїв).

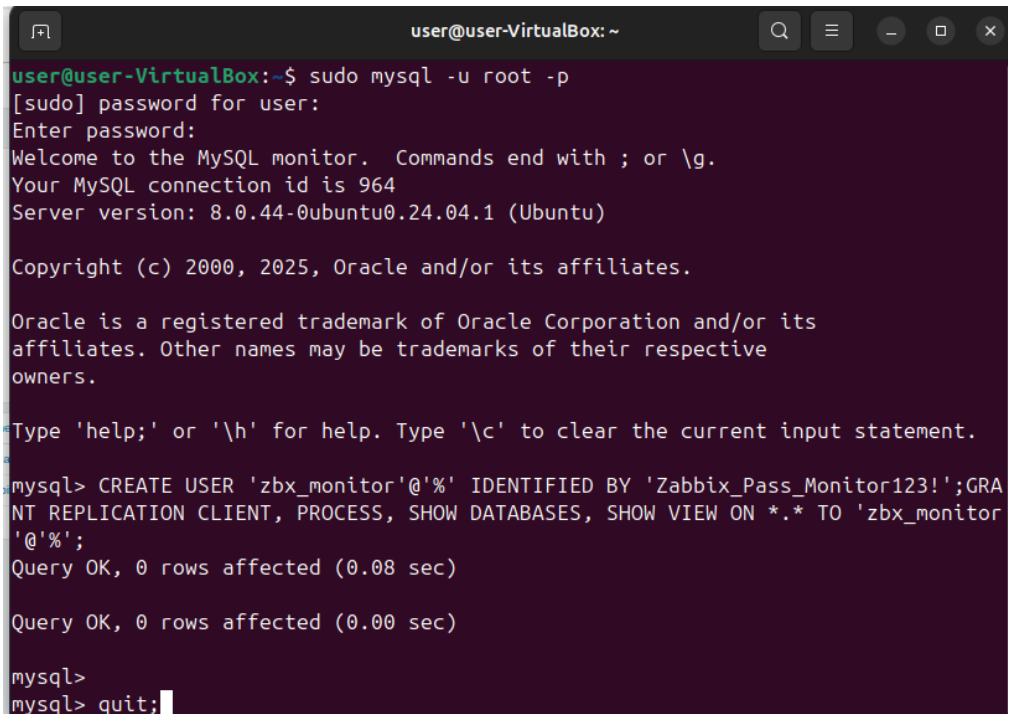


Рисунок 3.24 – Створення сервісного користувача для моніторингу

Розширення функціоналу моніторингу хоста Virtual-Node-01, рис.3.25. До існуючої конфігурації додається шаблон "MySQL by Zabbix agent 2". Цей шаблон містить наперед визначений набір правил (Items) та тригерів для аналізу швидкодії бази даних, статусу буфера InnoDB та кількості повільних запитів.

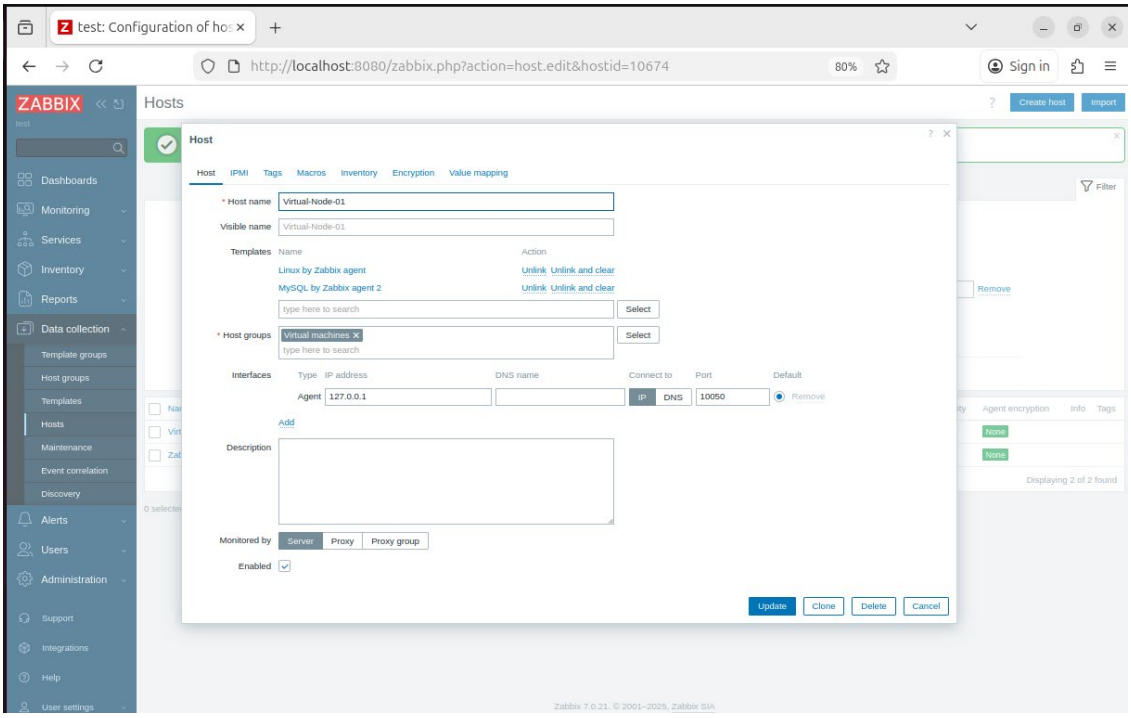


Рисунок 3.25 – Призначення спеціалізованого шаблону

Налаштування параметрів підключення через механізм користувацьких макросів, рис.3.26. У вкладці "Macros" для хоста задаються змінні, які використовуються шаблоном:

— {$MYSQL.DSN} — рядок підключення (Data Source Name), вказано як tcp://127.0.0.1:3306.

— {$MYSQL.USER} — ім'я користувача, створеного на етапі 1 (zbx_monitor).

— {$MYSQL.PASSWORD} — пароль доступу (Zabbix_Pass_Monitor123!).

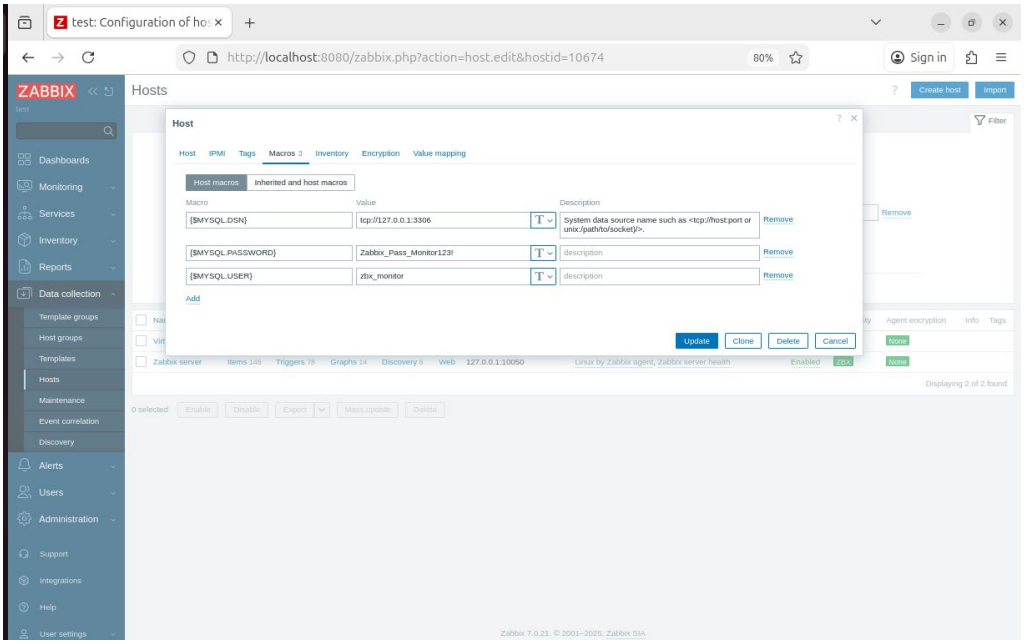


Рисунок 3.26 – Визначення макросів авторизації

Перегляд повної таблиці змінних шаблону, рис. 3.27.

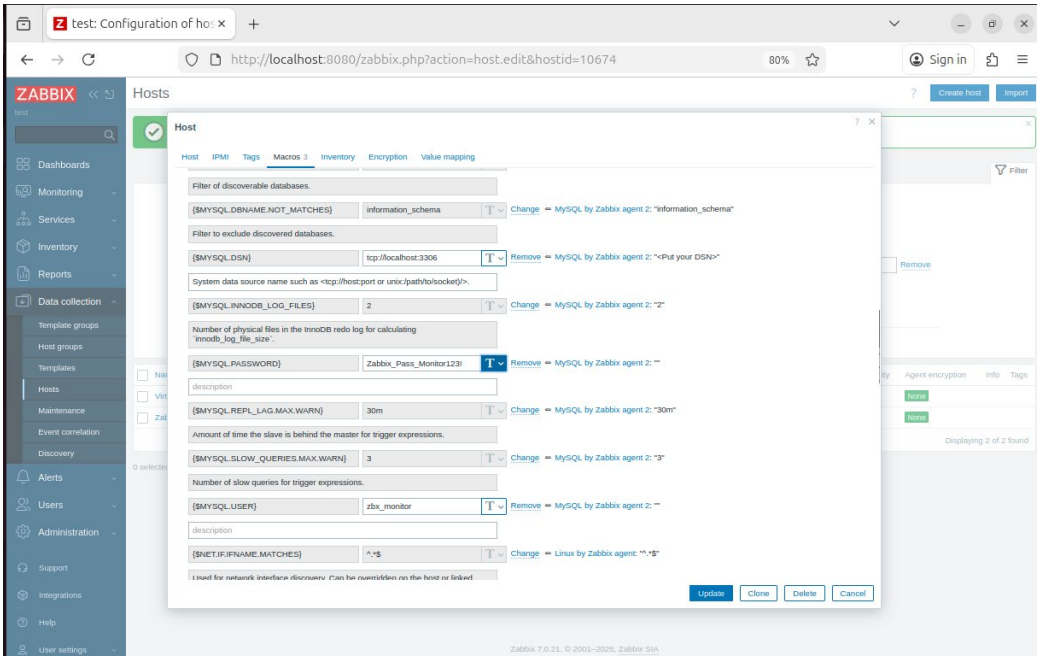


Рисунок 3.27 – Верифікація успадкованих та хостових макросів

Цей екран демонструє, як глобальні параметри шаблону, табл.3.1, поєднуються з локальними обліковими даними, введеними адміністратором.

Така ієрархія дозволяє гнучко керувати налаштуваннями моніторингу без необхідності редагування коду самого шаблону.

Таблиця 3.1 – Макроси шаблону моніторингу MySQL by Zabbix agent 2

Макрос	Значення за замовчуванням	Опис призначення
{$MYSQL.DSN}	tcp://localhost:3306	URI для підключення до бази даних (протокол, хост, порт).
{$MYSQL.USER}	zbx_monitor	Ім'я користувача для моніторингу, створеного в БД.
{$MYSQL.PASSWORD}	Zabbix_Pass_Monitor123!	Пароль користувача (прихований з міркувань безпеки).
{$MYSQL.ABORTED_CONN.MAX.WARN}	3	Кількість обірваних з'єднань, при якій спрацює тригер.
{$MYSQL.REPL_LAG.MAX.WARN}	30m	Максимально допустиме відставання реплікації (30 хвилин).
{$MYSQL.SLOW_QUERIES.MAX.WARN}	3	Кількість "повільних" запитів для генерації попередження.
{$MYSQL.BUFF_UTIL.MIN.WARN}	50	Мінімальний відсоток використання буфера InnoDB.

3.7 Верифікація системи моніторингу шляхом генерації синтетичного навантаження

Ключовим етапом впровадження системи моніторингу є перевірка її реактивності — здатності коректно фіксувати та візуалізувати зміни у стані

об'єкта спостереження в реальному часі. Для підтвердження працездатності налаштованих шаблонів (OS Linux та MySQL) було проведено процедуру стрес-тестування. Метою експерименту є створення штучного пікового навантаження на СУБД та аналіз відображення цих подій на графіках Zabbix.

Виконання команди mysqlslap в терміналі сервера, рис.3.28. Ця утиліта генерує синтетичне навантаження на базу даних, емулюючи одночасну роботу декількох клієнтів. Використані параметри: --concurrency=10 (10 одночасних потоків) та --iterations=50 (кількість повторень тесту). Успішне виконання запитів, що стало тригером для зміни метрик системи.

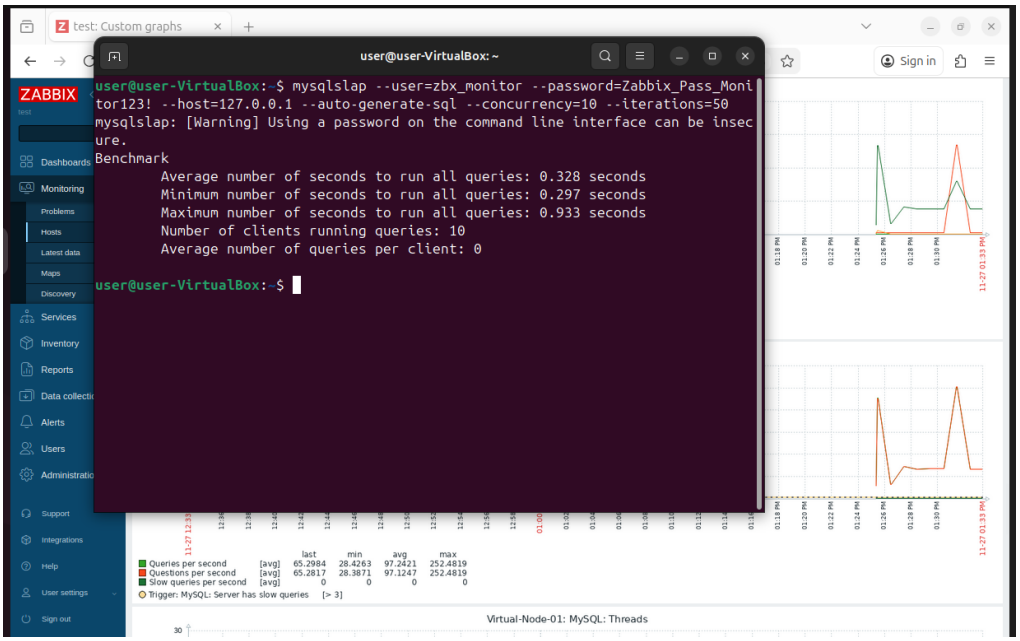


Рисунок 3.28 – Емуляція активності користувачів

Візуалізація впливу стрес-тесту на апаратні ресурси:

1. CPU utilization: На графіку чітко видно різкий сплеск (Spike) використання процесорного часу, що корелює з моментом запуску mysqlslap, рис.3.29.

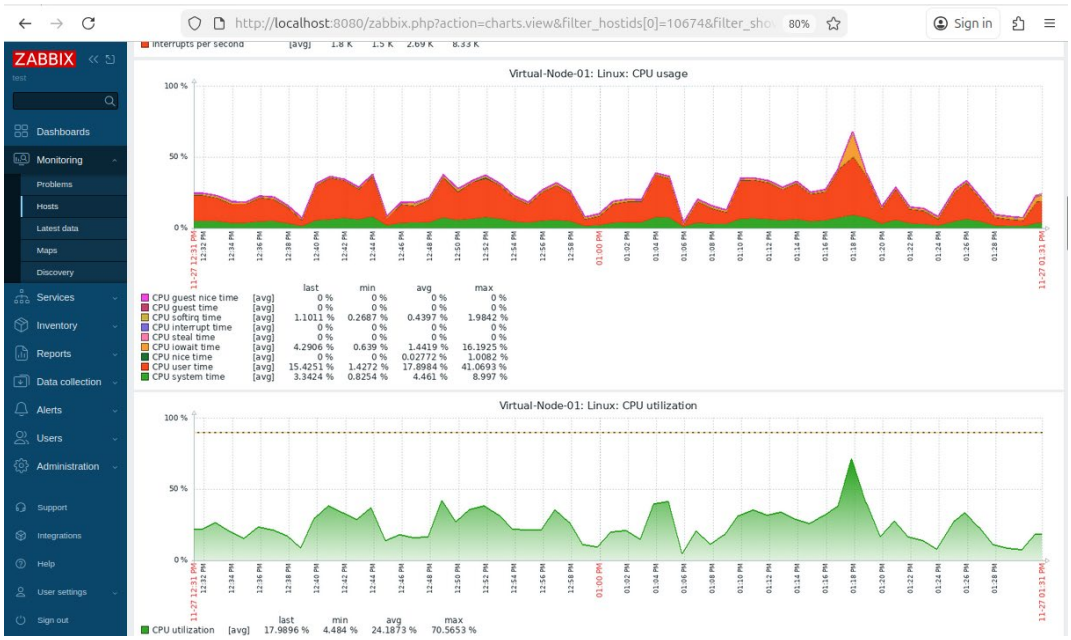


Рисунок 3.29 – Графік навантаження CPU під час тестування

2. CPU Jumps/Context switches: Зростання кількості перемикань контексту свідчить про інтенсивну обробку черги запитів ядром ОС, рис.3.30.

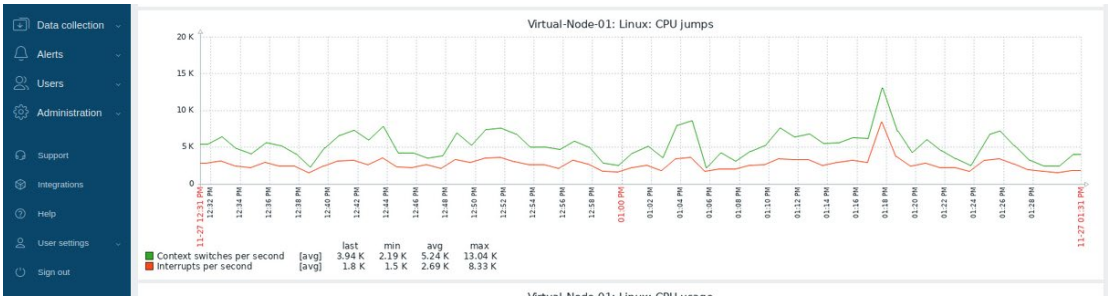


Рисунок 3.30 – Графік обробки черги запитів ядром ОС

3. System Load: Підвищення показника середнього завантаження системи (Load Average), що підтверджує наявність черги процесів на виконання, рис.3.31.

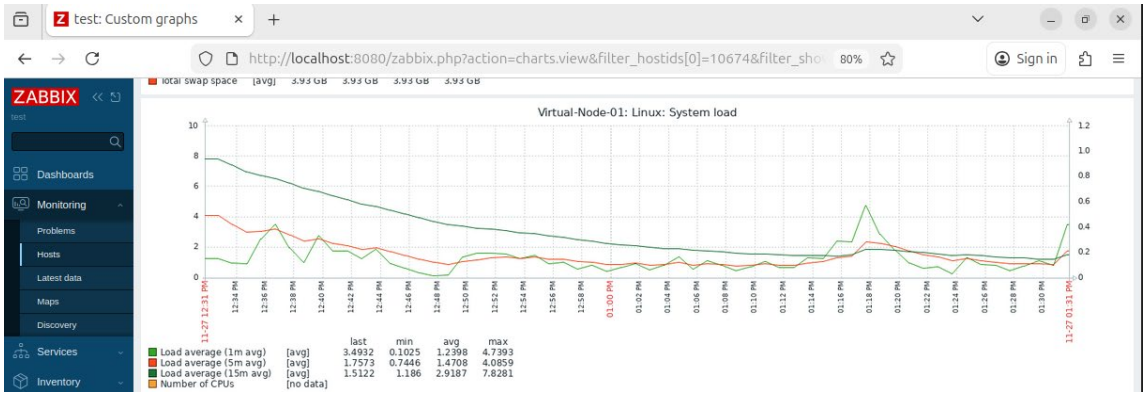


Рисунок 3.31 – Графік середнього завантаження системи

Детальний аналіз внутрішніх метрик бази даних, зібраних через Zabbix Agent 2. MySQL Operations/Queries: Графіки демонструють миттєве зростання кількості операцій SELECT та INSERT (до 250+ запитів на секунду) у момент тестування, рис.3.32.

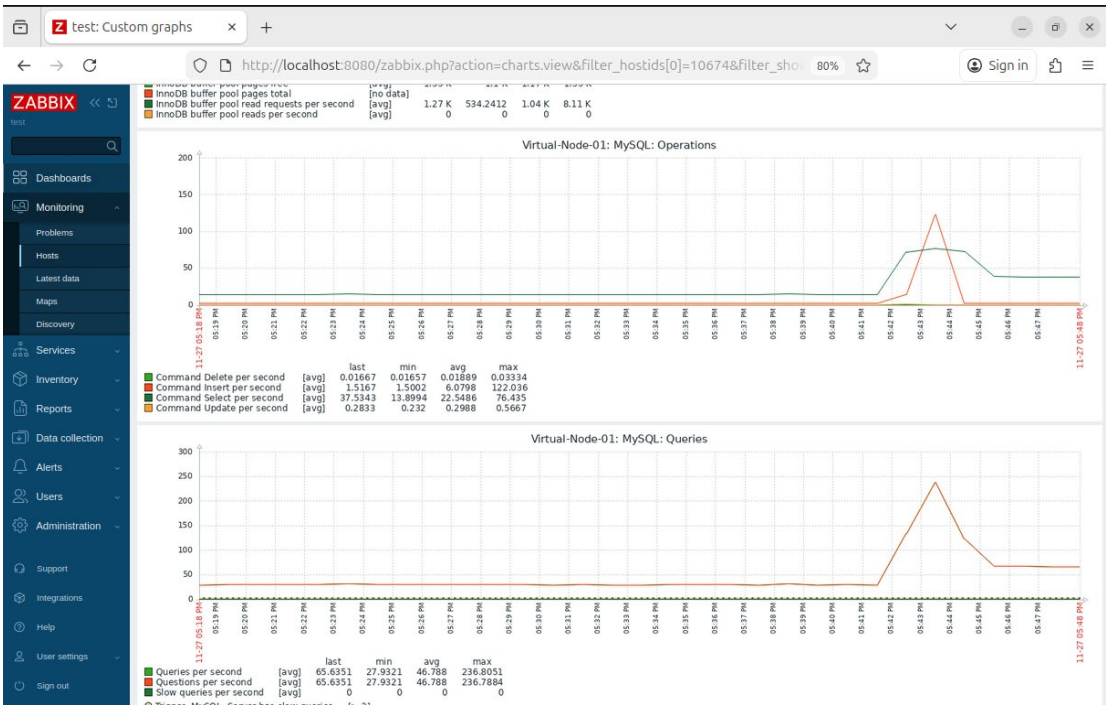


Рисунок 3.32 – Графік зміни навантаження MySQL Operations/Queries

MySQL Bandwidth: Фіксація збільшення обсягу переданих даних (трафіку), що генерується відповідями бази даних, рис.3.33.

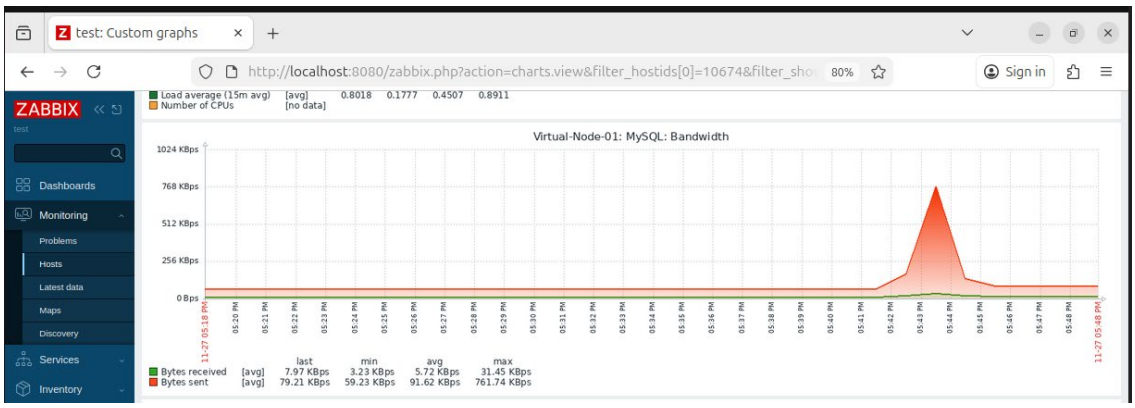


Рисунок 3.32 – Графік обсягу переданих даних

Connections: Реєстрація нових підключень до БД, ініційованих утилітою емуляції, рис.3.33.

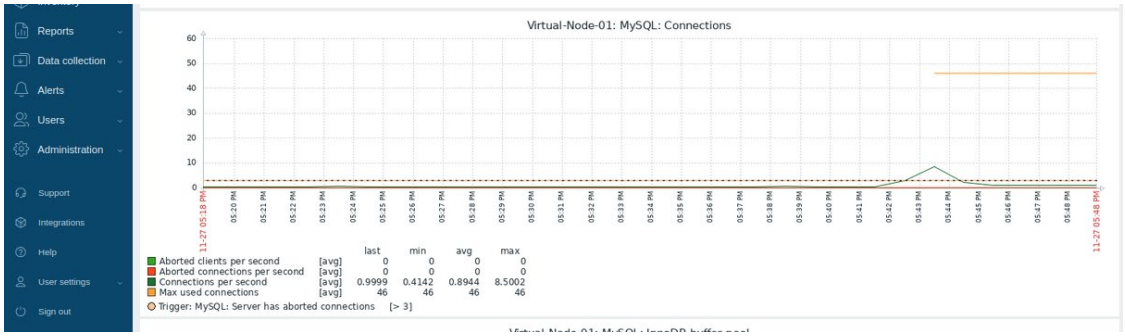


Рисунок 3.33 – Графік реєстрації нових підключень до БД

Моніторинг статичних ресурсів. Memory Usage: Графік демонструє стабільне споживання оперативної пам'яті, що вказує на те, що поточне навантаження не призвело до вичерпання RAM або активного використання Swap-розділу, рис. 3.34.

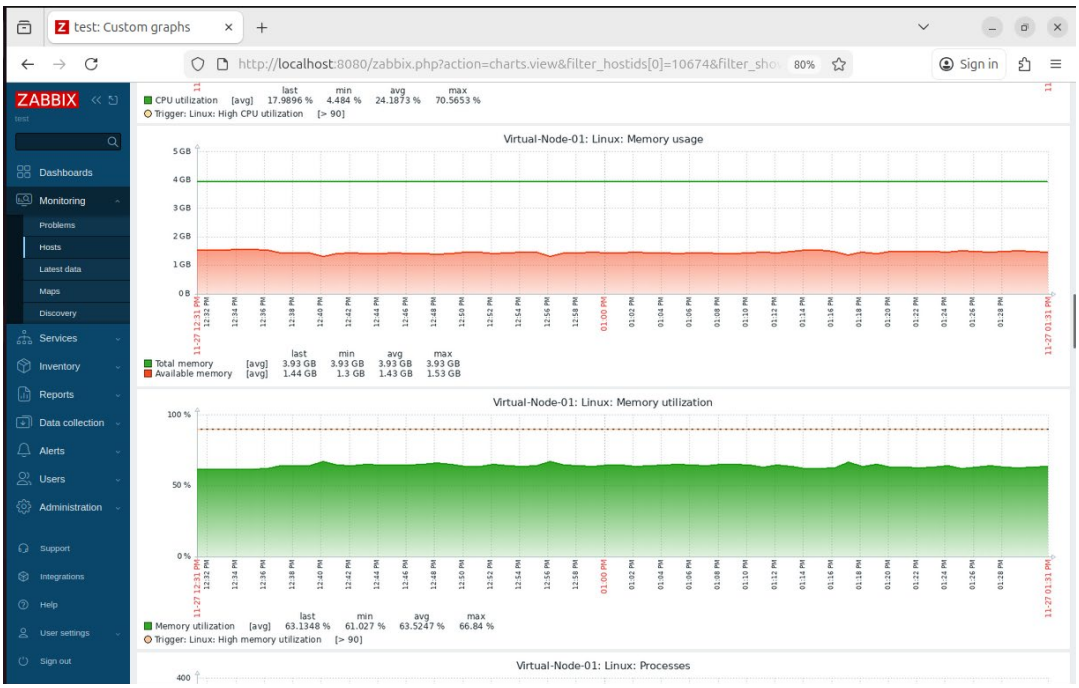


Рисунок 3.34 – Графік споживання оперативної пам'яті

File System: Контроль вільного місця на диску. Оскільки тест був короткостроковим, значного приросту зайнятого простору не зафіксовано, проте система продовжує відстежувати цей критичний параметр, рис.3.35.

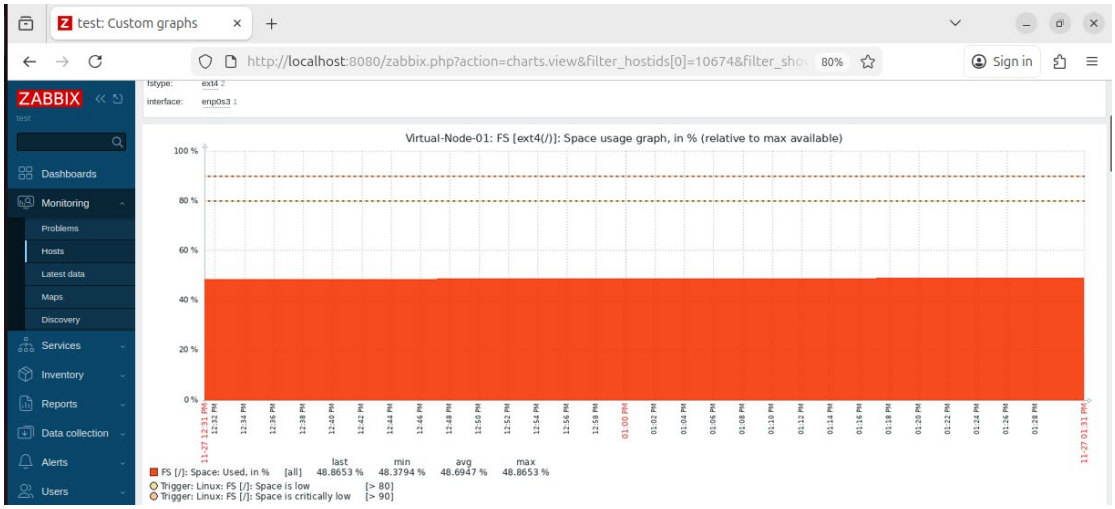


Рисунок 3.35 – Графік контролю вільного місця на диску

3.8 Діагностика аварійних станів та аналіз механізмів сповіщення про інциденти

Невід'ємною складовою експлуатації системи моніторингу є перевірка її здатності коректно ідентифікувати збої в роботі інфраструктури (Incident Detection) та автоматично фіксувати відновлення сервісів (Recovery Detection). У даному розділі проаналізовано поведінку Zabbix Server при виникненні типових проблем: недоступності прикладного сервісу, зміни конфігурації операційної системи та втраті мережевого зв'язку з агентом. Демонстрація роботи підсистеми тригерів, рис.3.36. Система зафіксувала дві події різного рівня критичності:

— "MySQL: Failed to fetch info data" (Рівень: Information) — вказує на те, що агент не зміг отримати метрики від СУБД (ймовірно, до моменту налаштування коректних макросів авторизації).

— "Linux: Number of installed packages has been changed" (Рівень: Warning) — реакція на встановлення нових програм (у даному випадку самого агента та утиліт), що є важливою частиною аудиту безпеки.

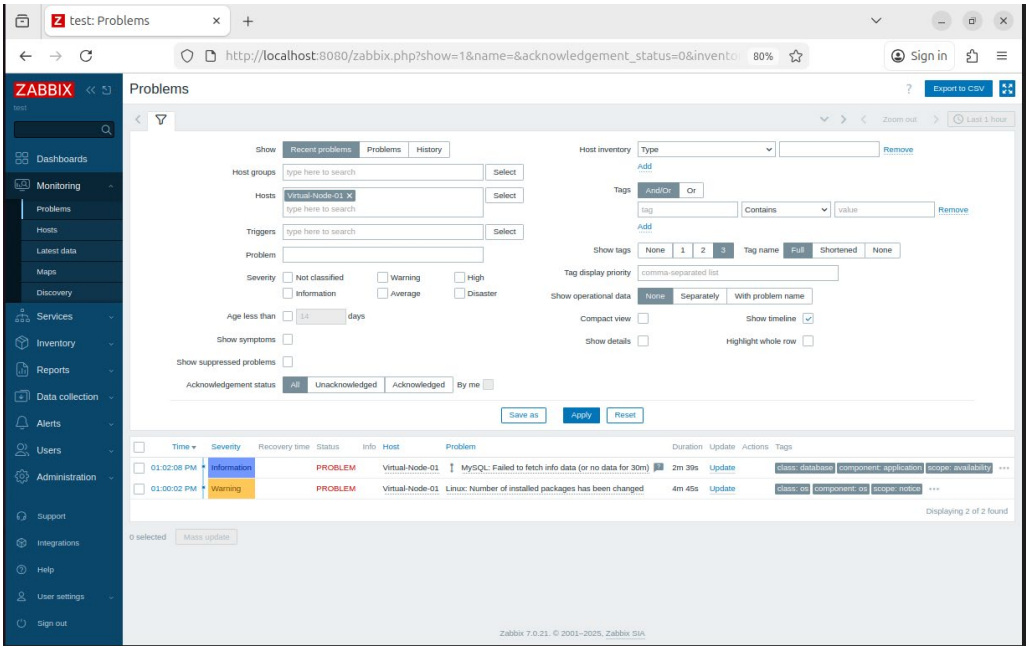


Рисунок 3.36 – Реєстрація активних тригерів у панелі "Problems"

Життєвий цикл інциденту. Статус проблеми з MySQL змінився на RESOLVED (зелений колір, час відновлення 01:24:30 PM), рис.3.37. Це демонструє здатність Zabbix автоматично закривати інциденти, як тільки умови тригера перестають виконуватися (тобто зв'язок з базою даних відновився), що зменшує навантаження на адміністратора.

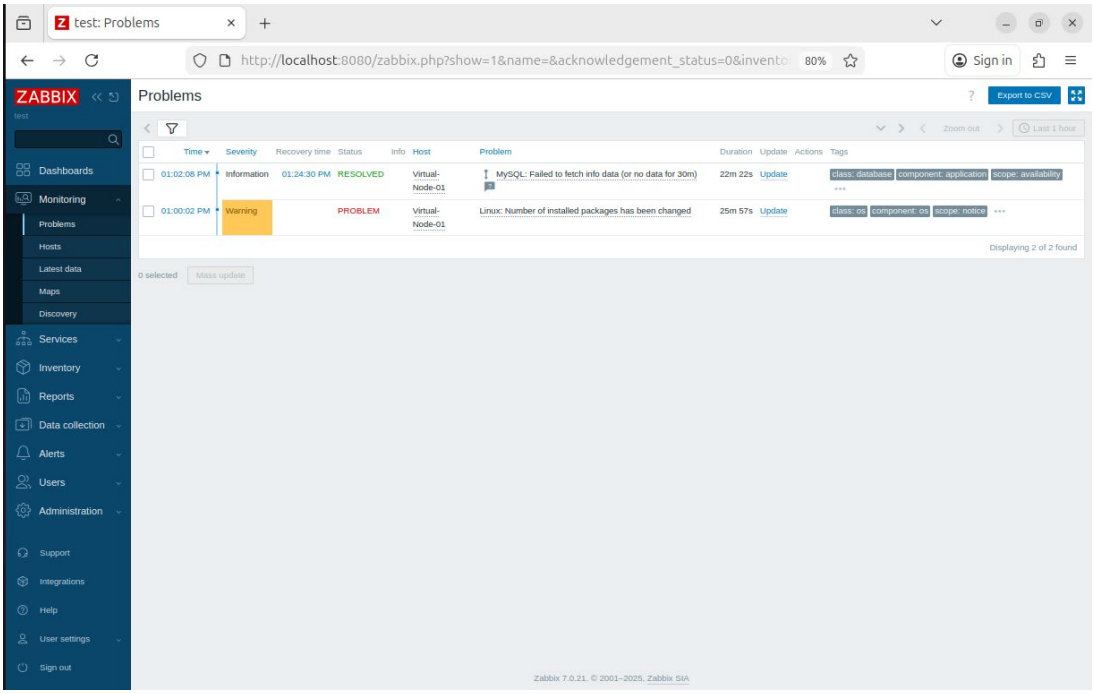


Рисунок 3.37 – Кореляція подій та автоматичне вирішення проблем

Симуляція повної недоступності хоста (наприклад, зупинка служби агента або обрив мережі). Індикатор ZBX змінив колір на червоний. Спливаюче повідомлення містить технічну причину збою: "Get value from agent failed: ... Connection refused". Це свідчить про те, що сервер намагався встановити TCP-з'єднання на порт 10050, але отримав відмову на рівні транспортного протоколу, рис.3.38.

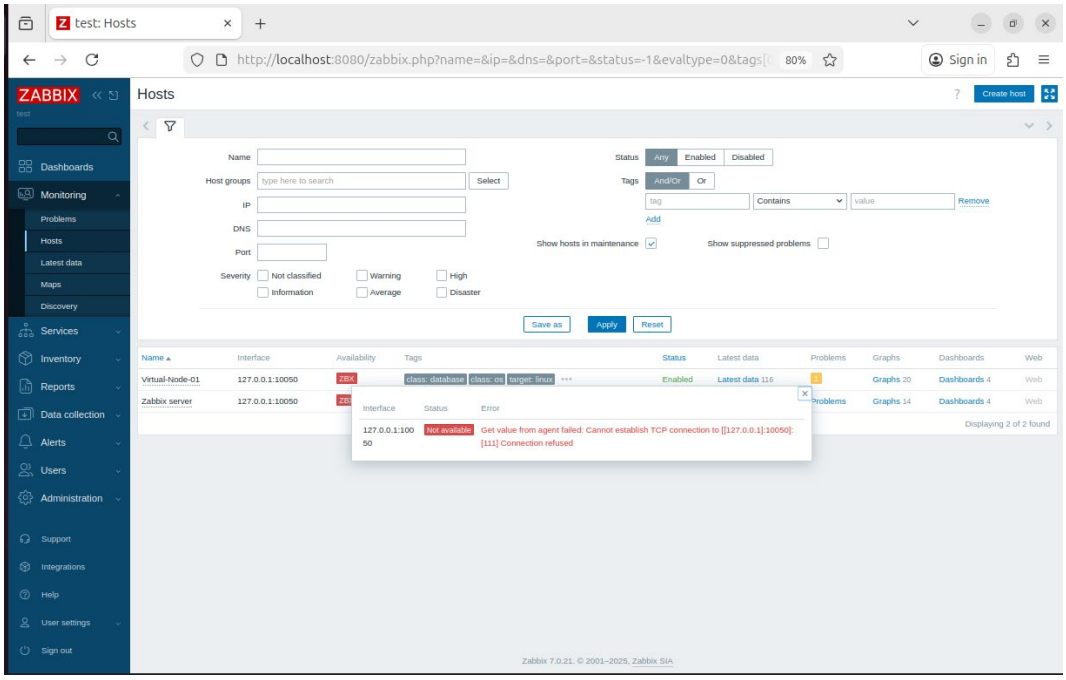


Рисунок 3.38 – Симуляція повної недоступності хоста

Підтвердження успішної реінтеграції хоста в контур моніторингу після усунення несправності. Індикатор доступності ZBX повернувся у зелений стан, а поле "Error" очистилося. Це означає, що канал передачі даних між Zabbix Server та Zabbix Agent 2 повністю відновлено, і збір метрик продовжується у штатному режимі, рис.3.39.

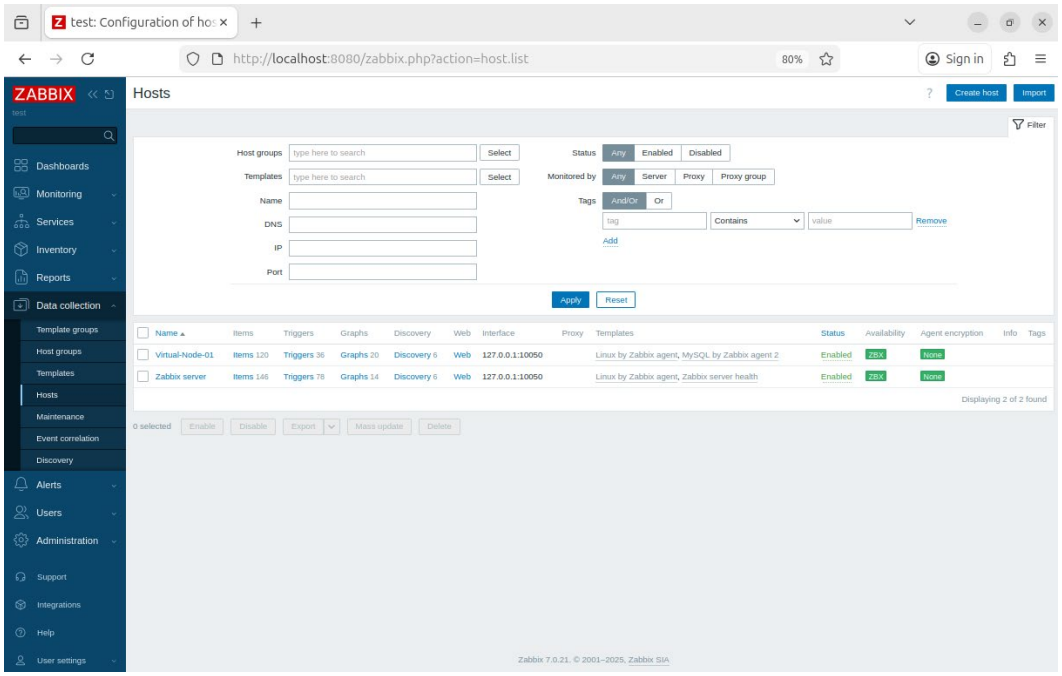


Рисунок 3.39 – Симуляція реінтеграції хоста в контур моніторингу

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання щодо забезпечення надійності віртуальних середовищ шляхом впровадження системи централізованого моніторингу.

Основні результати роботи полягають у наступному:

1. Встановлено, що для гетерогенних середовищ найбільш оптимальним є використання системи Zabbix версії 7.0 LTS. Вона перевершує аналоги завдяки відкритості, підтримці агентних та безагентних методів збору даних, а також потужним можливостям масштабування та інтеграції.

2. Успішно розгорнуто backend-частину системи на базі ОС Ubuntu 24.04 LTS. Використання зв’язки Nginx та PHP-FPM забезпечило високу продуктивність веб-інтерфейсу, а налаштування MySQL з параметрами безпеки гарантує цілісність збережених даних моніторингу.

3. Налаштовано міжмережевий екран з доступом лише до критичних портів, що мінімізує вектори атак на сервер моніторингу. Забезпечено коректну взаємодію компонентів через systemd для автоматичного відновлення роботи після перезавантаження.

4. Впровадження Zabbix Agent 2 дозволило реалізувати збір метрик у багатопоточному режимі. Налаштовано активні перевірки, що дозволяє агенту самостійно ініціювати передачу даних, знижуючи навантаження на центральний сервер.

5. Реалізовано глибокий моніторинг СУБД MySQL. Використання спеціалізованих шаблонів та макросів дозволило відстежувати не лише доступність сервісу, а й специфічні метрики: кількість повільних запитів, використання буфера InnoDB та пропускну здатність.

6. Проведене стрес-тестування за допомогою mysqlslap підтвердило реактивність системи. Графіки Zabbix миттєво відобразили сплески навантаження на CPU та збільшення кількості запитів, а система тригерів

коректно зафіксувала інциденти та їх автоматичне вирішення після стабілізації стану.

Результати тестування свідчать про підвищення ефективності адміністрування:

— Забезпечено 100% видимість стану підключених віртуальних вузлів.

— Час виявлення критичних проблем: недоступність хоста, зупинка сервісу БД, скорочено до автоматичного інтервалу опитування, менше 1 хвилини.

— Впроваджено механізм проактивного реагування на вичерпання ресурсів, що дозволяє запобігати аваріям до їх настання.

Таким чином, розроблена система є готовим інструментом для керування віртуальними середовищами, який гарантує стабільність надання IT-послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Catalin-Marian Petruti et al. "Automatic Management Solution in Cloud Using NtopNG and Zabbix." 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet) (2018): 1-6. https://doi.org/10.1109/roedunet.2018.8514142.

2. Jie Bai et al. "Design and implementation of small private cloud system with highly reliable management and monitoring." , 12287 (2022): 122870T - 122870T-5. https://doi.org/10.1117/12.2640748.

3. Tiangui Cao et al. "Application of Network Systems Management Cloud Service Based on Open Source Zabbix.", 07 (2018): 200-210. https://doi.org/10.12677/sea.2018.73023.

4. Mohamad Cholid Sefulloh et al. "Making a Web-Based Service Request Management System Application at The Financial and Development Supervisory Agency (BPKP)." Jurnal Teknologi Informatika dan Komputer (2025). https://doi.org/10.37012/jtik.v11i2.2755.

5. Jonathan Chua et al. "Building a Cloud Infrastructure for Virtual Machine Scheduling in Datacenters." 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC) (2024): 0105-0110. https://doi.org/10.1109/ccwc60891.2024.10427797.

6. Denna Irianto et al. "Design and Implementation of Network and Server Monitoring Using Zabbix at The Financial and Development Supervisory Agency." Jurnal Teknologi Informatika dan Komputer (2025). https://doi.org/10.37012/jtik.v11i2.2756.

7. S. Mescheryakov et al. "Adaptive control of cloud computing resources in the Internet telecommunication multiservice system." 2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) (2014): 287-293. https://doi.org/10.1109/icumt.2014.7002117.

8. Peixian Cong et al. "Research on Cluster Monitoring and Prediction Platform based on Zabbix Technology." IOP Conference Series: Earth and Environmental Science, 512 (2020). https://doi.org/10.1088/1755-1315/512/1/012155.

9. Verónica Castro Ayarza et al. "Cluster Monitoring and Integration in Technology Company." Advances in Intelligent Systems and Computing (2019). https://doi.org/10.1007/978-3-030-33547-2_19.

10. Mehmet Kara et al. "Efficiency of Adding a New Template to Network Management System and Zabbix Application." 2025 7th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (ICHORA) (2025): 1-6. https://doi.org/10.1109/ichora65333.2025.11017256.

11. Natalya Yu. Bumazhkina et al. "An approach to developing the architecture of a software package to support the process of redevelopment of virtual machines in a virtualized data center." University proceedings. Volga region. Technical sciences (2025). https://doi.org/10.21685/2072-3059-2025-1-2.

12. Muhammad Aliyul Husna et al. "Implementasi Sistem Monitoring Jaringan dan Server Menggunakan Zabbix yang Terintegrasi dengan Grafana dan Telegram." JURIKOM (Jurnal Riset Komputer) (2021). https://doi.org/10.30865/jurikom.v8i6.3631.

13. Arya Pradana et al. "Implementasi Sistem Monitoring Jaringan Menggunakan Zabbix Berbasis SNMP." AITI (2022). https://doi.org/10.24246/aiti.v19i2.248-262.

14. Mohd Faris Mohd Fuzi et al. "Integrated Network Monitoring using Zabbix with Push Notification via Telegram." Journal of Computing Research and Innovation (2022). https://doi.org/10.24191/jcrinn.v7i1.282.

15. Kota Suzuki et al. "Proposal for Server Integrated Management System Using Zabbix for Blockchain-Based NTMobile." 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE) (2020): 773-774. https://doi.org/10.1109/gcce50665.2020.9291802.

16. Mit Yogeshkumar Pandya et al. "Uptime Monitoring and Reporting Strategies: A Case Study." International Journal for Research in Applied Science and Engineering Technology (2024). https://doi.org/10.22214/ijraset.2024.59855.

17. Sulasno Sulasno et al. "Developing Integrated Smartphones Notification of Server Resource Monitoring System Using Zabbix, Webhook, and Telegram." JUITA: Jurnal Informatika (2021). https://doi.org/10.30595/juita.v9i2.10411.

18. et al. "Network monitoring tools: enabling the management of network assets, ensuring high availability." International Journal of Development Research (2020). https://doi.org/10.37118/ijdr.19224.06.2020.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)