Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«АВТОМАТИЗОВАНА СИСТЕМА ДЛЯ ВИЯВЛЕННЯ ПРОБЛЕМ ЗА ДОПОМОГОЮ СИСТЕМИ МОНІТОРИНГУ ZABBIX»

Виконав: студент IV курсу, групи 1АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Максим КУПРІЄНКО
(прізвище та ініціали)
Керівник: д.т.н.,професор кафедри АІІТ
Роман КВЄТНИЙ
(прізвище та ініціали)
Консультант: к.т.н.,доцент кафедри АІІТ
Ілона БОГАЧ
(прізвище та ініціали)
«16» 06 2025 р.
Рецензент: д.т.н., професор кафедри КН
Ярослав ІВАНЧУК
(прізвище та ініціали)
«16» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
Олег БІСІКАЛО
«16» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 15 –Автоматизація та приладобудування
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
_____ д.т.н., проф. Олег БІСІКАЛО
«16» 06 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Купрієнку Максиму Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизована система для виявлення проблем за допомогою системи моніторингу Zabbix»
керівник роботи Квєтний Роман Наумович, д.т.н, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи:
- використання системи Zabbix для моніторингу IT-інфраструктури;
- моніторинг щонайменше 6 різних типів пристроїв або сервісів (сервери, мережеві пристрої, додатки тощо);
- використання протоколів SNMP, IPMI, WMI або API для збору даних;
- налаштування системи сповіщень через Telegram-бот для оперативного інформування про інциденти.
4. Зміст текстової частини (перелік питань, які потрібно розробити) Аналіз предметної області. Дослідження системи моніторингу з використанням ZABBIX. Практична реалізація системи. Тестування розгорнутої системи.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) Схема архітектури автоматизованої системи моніторингу на базі Zabbix, схема конфігурації тригерів та сповіщень у системі Zabbix, діаграма роботи Telegram-бота для сповіщень про інциденти, приклад дашборду Zabbix для візуалізації метрик моніторингу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Богач І.В, доц. каф. АПТ		

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	03.10.2024	09.10.2024	
2	Аналіз предметної області та опрацювання літературних джерел.	12.03.2025	29.03.2025	
3	Постановка задач для вирішення в БКР	01.04.2025	26.04.2025	
4	Вирішення поставлених задач	29.04.2025	10.05.2025	
5	Підтвердження достовірності отриманих результатів	13.05.2025	24.05.2025	
6	Оформлення пояснювальної записки та графічної частини	27.05.2025	31.05.2025	
7	Перевірка ПЗ на відповідність вимогам	03.06.2025	07.06.2025	
8	Попередній захист, доопрацювання, та рецензування БКР	10.06.2025	15.06.2025	
9	Захист БКР	16.06.2025	23.06.2025	

Студент _____ Максим КУПРІЄНКО
(підпис) (ім'я та прізвище)

Керівник роботи _____ Роман КВЄТНИЙ
(підпис) (ім'я та прізвище)

Дана бака
А4, в тому числ
джерел містить
Дана роб
проблем за допо
У роботі
розгортання й
аналізу було
версії 7.2, яка
інфраструкту
гіпервізора К
забезпечує
адмініструва
Ключ

АНОТАЦІЯ

Дана бакалаврська кваліфікаційна робота складається з 94 сторінок формату А4, в тому числі додатків, на яких є 11 рисунків, 2 таблиці, список використаних джерел містить 22 найменувань.

Дана робота присвячена розробці автоматизованої системи для виявлення проблем за допомогою системи моніторингу Zabbix.

У роботі проведено аналіз предметної області та досліджено особливості розгортання й використання систем моніторингу. На основі отриманих знань і аналізу було спроєктовано та реалізовано автоматизовану систему на базі Zabbix версії 7.2, яка включає компоненти для збору, обробки та візуалізації даних про стан інфраструктури. Система розгорнута у віртуалізованому середовищі на основі гіпервізора KVM з операційною системою Debian 12 та базою даних MySQL, що забезпечує ефективне виявлення проблем, моніторинг ресурсів і оптимізацію адміністрування.

Ключові слова: моніторинг, Zabbix, автоматизація, безпека, Telegram.

ABSTRACT

This bachelor's thesis consists of 94 A4 pages, including appendices, which contain 11 figures, 2 tables, and a reference list with 22 sources.

This work is dedicated to the development of an automated system for problem detection using the Zabbix monitoring system.

The study includes an analysis of the subject area and an investigation of the specifics of deploying and utilizing monitoring systems. Based on the acquired knowledge and analysis, an automated system was designed and implemented using Zabbix version 7.2, which incorporates components for collecting, processing, and visualizing infrastructure status data. The system is deployed in a virtualized environment based on the KVM hypervisor with the Debian 12 operating system and MySQL database, ensuring efficient problem detection, resource monitoring, and administration optimization.

Keywords: monitoring, Zabbix, automation, security, Telegram.

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасному світі, де IT-інфраструктура є основою для функціонування більшості бізнес-процесів, забезпечення її стабільності та надійності набуває критичного значення. Автоматизовані системи моніторингу, такі як Zabbix, відіграють ключову роль у виявленні та запобіганні збоям, що дозволяє підвищити ефективність роботи адміністраторів, знизити ризики простоїв і забезпечити безперервність бізнес-процесів.

Впровадження Zabbix сприяє автоматизації моніторингу серверів, мережевих пристроїв і додатків, а також забезпечує оперативне сповіщення про інциденти та аналіз даних. Це дозволяє оптимізувати роботу IT-відділів, скоротити час на усунення проблем і підвищити загальну надійність IT-сервісів.

Проте процес впровадження таких систем супроводжується певними викликами, зокрема необхідністю адаптації до потреб конкретної організації, навчання персоналу, забезпечення безпеки даних та інтеграції з іншими системами. Незважаючи на ці труднощі, перспективи розвитку автоматизованих систем моніторингу є надзвичайно великими. Сучасні технології, такі як штучний інтелект, хмарні рішення та автоматизація аналітики великих даних, відкривають нові можливості для покращення якості моніторингу та прогнозування збоїв.

Актуальним є розробка автоматизованої системи моніторингу з метою підвищення надійності IT-інфраструктури, оптимізації роботи адміністраторів та забезпечення безперервності бізнес-процесів.

Мета роботи полягає в удосконаленні сервісів моніторингу IT інфраструктури.

Для досягнення поставленої мети необхідно **вирішити наступні задачі:**

1. Провести аналіз предметної галузі
2. Дослідити основні переваги та виклики впровадження Zabbix у різних організаціях.
3. Оцінити перспективи розвитку систем моніторингу, зокрема впровадження штучного інтелекту, хмарних технологій та автоматизації аналітики.

4. Розгорнути та налаштувати систему.

5. Протестувати розгорнуту систему та порівняти з аналогами.

Об’єктом дослідження є процеси моніторингу IT-інфраструктури із застосуванням автоматизованих систем.

Предметом дослідження є методи та технології розгортання та використання системи Zabbix для моніторингу IT-інфраструктури.

Методи дослідження. У роботі застосовуються методи аналізу та синтезу інформації, порівняльний аналіз функціональних можливостей систем моніторингу, моделювання процесів моніторингу, а також методи оцінки ефективності автоматизації.

Науково-практичний результат роботи полягає у розробці вдосконаленої автоматизованої системи моніторингу IT інфраструктури з використанням Zabbix, що на відміну від існуючих розширять функціонал системи, дозволяючи виявляти потенційні кібератаки та апаратні збої. більшу надійність та безперервну роботу бізнес-процесів.

Апробація результатів роботи. Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025) [1] та на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [21].

Впровадження результатів роботи. Результати роботи планується використати в розробках та управлінні проєктами ТОВ «Спільна справа» (додаток Д).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття моніторингу IT-інфраструктури

У сучасному цифровому світі IT-інфраструктура є основою функціонування більшості організацій, незалежно від їх розміру чи сфери діяльності. Вона охоплює апаратне забезпечення (сервери, мережеві пристрої, системи зберігання даних), програмне забезпечення (операційні системи, бази даних, веб-сервіси) та мережеві компоненти, які забезпечують безперебійну роботу бізнес-процесів[11]. Від електронного документообігу до веб-сервісів, від внутрішніх систем управління до хмарних платформ — усе це залежить від стабільності та надійності IT-інфраструктури. Будь-які збої, затримки або порушення в роботі IT-систем можуть призвести до серйозних наслідків, таких як фінансові втрати, зниження продуктивності, втрата довіри клієнтів або навіть репутаційні ризики.

Моніторинг IT-інфраструктури є процесом безперервного спостереження за станом усіх компонентів системи з метою виявлення відхилень від нормального функціонування, попередження потенційних збоїв і забезпечення стабільної роботи[12][13]. Цей процес включає збір даних про ключові параметри (метрики), їх аналіз, візуалізацію та, за необхідності, автоматичне або ручне реагування на виявлені проблеми. Метрики можуть охоплювати широкий спектр показників: завантаженість процесора, використання оперативної пам’яті, обсяг вільного місця на дисках, затримки в мережевому трафіку, доступність веб-сервісів, час відгуку додатків тощо[19].

Моніторинг IT-інфраструктури відіграє ключову роль у забезпеченні безперервності бізнес-процесів. Наприклад, у сфері електронної комерції навіть короткочасний простій веб-сайту може призвести до втрати продажів, тоді як у фінансовому секторі збої в IT-системах можуть спричинити порушення транзакцій або втрату даних. Таким чином, ефективний моніторинг є не просто технічною необхідністю, а стратегічним інструментом для підтримки конкуренто-спроможності організації.

Розглянемо рівні моніторингу IT-інфраструктури. Моніторинг IT-інфраструктури поділяється на кілька рівнів, кожен з яких відповідає за контроль певного аспекту системи. Ці рівні взаємодіють між собою, створюючи цілісну картину стану IT-середовища. Нижче наведено основні рівні моніторингу:

1. *Інфраструктурний моніторинг.* Цей рівень зосереджений на контролі фізичних і віртуальних компонентів IT-інфраструктури, таких як сервери, мережеві комутатори, маршрутизатори, джерела безперебійного живлення (UPS) та системи охолодження. Основні метрики включають температуру обладнання, стан вентиляторів, споживання електроенергії та доступність пристроїв. Наприклад, якщо сервер перегрівається через несправність системи охолодження, інфраструктурний моніторинг може виявити цю проблему до того, як вона призведе до апаратного збою.

2. *Системний моніторинг.* Системний моніторинг спрямований на спостереження за операційними системами (Windows, Linux, тощо) та їхніми службами. Він охоплює контроль таких параметрів, як використання процесора (CPU), оперативної пам’яті (RAM), дискового простору, а також стан системних процесів і служб. Наприклад, системний моніторинг може виявити, що служба бази даних MySQL припинила роботу, і сповістити адміністратора для оперативного реагування.

3. *Мережевий моніторинг.* Цей рівень відповідає за контроль мережевої інфраструктури, включаючи пропускну здатність каналів, втрату пакетів, затримки (latency) та доступність хостів. Мережевий моніторинг дозволяє виявляти проблеми, такі як перевантаження мережевих каналів або відмова маршрутизатора. Наприклад, якщо мережевий трафік між двома офісами компанії різко зростає, це може вказувати на неавторизовану активність або необхідність розширення пропускної здатності.

4. *Моніторинг додатків.* Моніторинг додатків зосереджений на перевірці працездатності програмного забезпечення, такого як веб-сервери (Apache, Nginx), бази даних (PostgreSQL, Oracle), CRM-системи чи корпоративні додатки. Ключові метрики включають час відгуку додатків, кількість помилок, продуктивність

запитів до бази даних тощо. Наприклад, моніторинг може виявити, що веб-сайт компанії видає помилку 503 (Service Unavailable) через перевантаження сервера.

5. *Моніторинг користувацького досвіду (UX)*. Цей рівень передбачає імітацію дій кінцевих користувачів для оцінки якості їхньої взаємодії з IT-сервісами. Наприклад, синтетичний моніторинг може періодично надсилати запити до веб-сайту, щоб перевірити час завантаження сторінки або доступність певних функцій. Якщо користувачі скаржаться на повільну роботу системи, UX-моніторинг допомагає виявити, чи проблема пов’язана з сервером, мережею чи самим додатком.

Кожен із цих рівнів має свої особливості, але разом вони створюють комплексну систему моніторингу, яка дозволяє охопити всі аспекти IT-інфраструктури. Важливо, щоб моніторинг був гнучким і адаптивним, оскільки сучасні IT-середовища часто включають гібридні інфраструктури (локальні сервери, хмарні платформи, контейнеризовані додатки).

Цілі та завдання моніторингу

Основна мета моніторингу IT-інфраструктури полягає в забезпеченні безперебійної роботи систем і мінімізації ризиків збоїв. Для досягнення цієї мети моніторинг виконує низку завдань:

- Раннє виявлення проблем: Моніторинг дозволяє виявляти відхилення від нормального стану системи (наприклад, зростання затримок у мережі) до того, як вони переростуть у серйозні інциденти.
- Превентивне реагування: Аналізуючи тренди та історичні дані, моніторинг допомагає передбачати потенційні проблеми, такі як нестача дискового простору або перевантаження сервера.
- Оптимізація ресурсів: Моніторинг виявляє "вузькі місця" в інфраструктурі, що дозволяє адміністраторам ефективніше розподіляти ресурси, наприклад, додавати пам’ять або масштабувати сервер.
- Забезпечення відповідності SLA: Моніторинг допомагає відстежувати показники, які впливають на угоди про рівень обслуговування (SLA), такі як час безвідмовної роботи (uptime) або час реагування на запити.

- Прозорість і звітність: Моніторинг забезпечує прозорість роботи IT-відділу, надаючи звіти про продуктивність систем, що є важливим для менеджменту та клієнтів.

Для виконання цих завдань моніторинг має бути не лише реактивним (фіксація вже наявних проблем), а й проактивним, тобто спрямованим на попередження інцидентів. Наприклад, якщо моніторинг показує, що дисковий простір на сервері зменшується швидше, ніж очікувалося, адміністратор може заздалегідь очистити непотрібні файли або додати додатковий диск.

Розглянемо роль моніторингу в ITSM. Моніторинг IT-інфраструктури є невід'ємною частиною управління IT-послугами (IT Service Management, ITSM), яке базується на стандартах, таких як ITIL (Information Technology Infrastructure Library). У контексті ITSM моніторинг виконує кілька ключових функцій:

1. Підтримка безперервності послуг: Моніторинг забезпечує постійний контроль за станом IT-послуг, що дозволяє швидко реагувати на інциденти та мінімізувати їхній вплив на бізнес.

2. Управління інцидентами: Моніторинг автоматично виявляє інциденти (наприклад, відмова сервера) і передає інформацію до системи управління інцидентами, що прискорює їх вирішення.

3. Планування потужностей: Аналіз даних моніторингу допомагає прогнозувати потреби в ресурсах, наприклад, необхідність у додаткових серверах або пропускній здатності мережі.

4. Забезпечення відповідності SLA: Моніторинг відстежує ключові показники ефективності (KPI), такі як доступність сервісів (availability) або час відновлення після збою (MTTR), що є основою для виконання SLA.

5. Покращення якості послуг: На основі зібраних даних організації можуть оптимізувати свої IT-процеси, наприклад, зменшувати час відповіді додатків або підвищувати стабільність систем.

Таким чином, моніторинг є основою для ефективного управління IT-послугами, забезпечуючи прозорість, обґрунтованість рішень і підвищення якості сервісів.

Розглянемо основні виклики сучасного моніторингу. Зі зростанням складності IT-систем моніторинг стикається з новими викликами. Сучасні IT-інфраструктури часто є гібридними, поєднуючи локальні сервери, хмарні платформи (AWS, Azure, Google Cloud), контейнери (Docker, Kubernetes) і мікросервісні архітектури[20]. Це створює такі проблеми:

- Великі обсяги даних: Сучасні системи генерують величезну кількість метрик, що потребує ефективних засобів їх обробки та зберігання.
- Динамічність середовища: У хмарних і контейнеризованих середовищах ресурси постійно створюються, знищуються або масштабуються, що ускладнює відстеження.
- Різноманітність технологій: Моніторинг має підтримувати різні протоколи (SNMP, WMI, API), операційні системи та типи пристроїв.
- Безпека: Зібрані дані можуть містити конфіденційну інформацію, тому моніторинг має забезпечувати шифрування та контроль доступу.
- Автоматизація: Ручний моніторинг стає неефективним через складність і масштаб сучасних систем, що вимагає автоматизованих рішень.

Для вирішення цих викликів використовуються автоматизовані системи моніторингу, такі як Zabbix, Nagios, Prometheus, Icinga та інші. Ці системи дозволяють централізовано збирати дані, налаштовувати тригери для виявлення проблем, генерувати сповіщення та візуалізувати метрики у зручному вигляді.

Автоматизація моніторингу

Автоматизація є ключовим елементом сучасного моніторингу IT-інфраструктури. Без неї адміністратори не в змозі оперативно обробляти великі обсяги даних або реагувати на інциденти в реальному часі. Автоматизовані системи моніторингу, такі як Zabbix, надають такі можливості:

- Автоматичний збір даних: Системи періодично опитують пристрої та сервіси, використовуючи протоколи SNMP, IPMI, WMI або API, без необхідності ручного втручання.
- Налаштування тригерів: Логічні умови (наприклад, “якщо CPU завантажено на 90% протягом 5 хвилин”) дозволяють автоматично виявляти проблеми.

- Сповіщення в реальному часі: Системи можуть надсилати повідомлення через email, Telegram, Slack або SMS, інформуючи адміністраторів про інциденти.
- Автоматичні дії: У разі виявлення проблеми система може виконувати заздалегідь визначені дії, наприклад, перезапускати службу або перенаправляти трафік на резервний сервер.
- Аналітика та звіти: Автоматизовані системи генерують звіти про продуктивність, тренди та інциденти, що полегшує планування та прийняття рішень.

Наприклад, система Zabbix підтримує як агентний, так і безагентний моніторинг, дозволяючи гнучко адаптуватися до різних середовищ. Вона може автоматично виявляти нові пристрої в мережі (Low-Level Discovery), налаштовувати шаблони моніторингу та інтегруватися з зовнішніми системами для сповіщень[2].

1.2 Аналіз основних проблем стабільності IT-систем

Стабільність IT-систем є ключовим фактором для забезпечення безперервної роботи сучасних організацій, які значною мірою залежать від інформаційних технологій. Проте в процесі експлуатації IT-інфраструктури виникають численні технічні, організаційні та зовнішні проблеми, які можуть призвести до збоїв, зниження продуктивності або навіть повної зупинки бізнес-процесів. Ці проблеми варіюються від апаратних несправностей до людських помилок і кібератак. У цьому розділі детально розглянуто основні категорії проблем стабільності IT-систем, їх причини, наслідки та способи їхнього попередження за допомогою ефективного моніторингу.

До основних проблем стабільності IT систем відносять:

1. Збої апаратного забезпечення
2. Перевантаження ресурсів
3. Мережеві збої.
4. Помилки програмного забезпечення
5. Людський фактор

6. Атаки та шкідливе програмне забезпечення

7. Значення своєчасного виявлення проблем

Розглянемо кожен з них більш детально:

1. Збої апаратного забезпечення. Однією з найпоширеніших причин нестабільності IT-систем є вихід із ладу апаратного забезпечення. Сервери, жорсткі диски, мережеві комутатори, маршрутизатори, джерела безперебійного живлення (UPS) та інші компоненти фізичної інфраструктури мають обмежений термін служби і можуть зазнавати поломок через знос, перегрів, перебої в електроживленні або виробничі дефекти.

- Приклади збоїв:

  - Несправність жорсткого диска в RAID-масиві може призвести до втрати даних або зниження продуктивності сервера.

  - Перегрів процесора через несправність системи охолодження може спричинити автоматичне відключення сервера для запобігання пошкодженню.

  - Відмова мережевого комутатора може ізолювати цілу підмережу, унеможлививши доступ до сервісів.

- Наслідки: Збої апаратного забезпечення часто призводять до простоїв (downtime), втрати даних або необхідності термінового ремонту чи заміни обладнання, що може бути дорогим і тривалим процесом. Наприклад, у сфері електронної комерції навіть кілька хвилин простою можуть коштувати компанії значних фінансових втрат через втрачені продажі.

- Роль моніторингу: Системи моніторингу, такі як Zabbix, дозволяють відстежувати стан апаратного забезпечення в реальному часі. Наприклад, датчики температури, показники SMART для жорстких дисків або статус живлення UPS можуть бути інтегровані в систему моніторингу. Тригери, налаштовані на критичні значення (наприклад, температура процесора вище 80°C), дозволяють адміністраторам діяти превентивно, замінюючи компоненти до їхньої повної відмови.

2. *Перевантаження ресурсів.* Перевантаження ресурсів є ще однією поширеною проблемою, яка виникає, коли попит на обчислювальні ресурси

перевищує їхню доступну кількість. Це може стосуватися процесора (CPU), оперативної пам’яті (RAM), дискового простору або пропускної здатності мережі.

- Приклади перевантаження:
  - Надмірне використання CPU через неправильно оптимізований додаток, який виконує ресурсоємні обчислення.
  - Переповнення дискового простору через накопичення лог-файлів або баз даних, що швидко зростають.
  - Нестача оперативної пам’яті внаслідок одночасної роботи кількох віртуальних машин на одному сервері.
- Наслідки: Перевантаження ресурсів може призвести до зниження продуктивності сервісів, затримок у виконанні запитів або навіть повного припинення роботи додатків. Наприклад, якщо база даних переповнює дисковий простір, це може зупинити транзакції в системі управління складом, що вплине на логістичні процеси компанії.
- Роль моніторингу: Системи моніторингу дозволяють відстежувати використання ресурсів у реальному часі та налаштовувати тригери для попередження про перевищення порогових значень. Наприклад, Zabbix може сповістити адміністратора, якщо використання CPU перевищує 90% протягом 5 хвилин або якщо вільний дисковий простір падає нижче 10%. Це дозволяє оперативно вжити заходів, таких як оптимізація додатків, очищення дисків або масштабування ресурсів.

3. *Мережеві збої.* Мережеві збої виникають через проблеми з доступністю вузлів, втрату зв’язку, перевантаження каналів або несправність мережевого обладнання, такого як маршрутизатори чи комутатори[16].

- Приклади мережевих збоїв:
  - Втрата зв’язку з віддаленим сервером через обрив кабелю або збій у роботі провайдера.
  - Перевантаження мережевого каналу через надмірний трафік, наприклад, під час пікового навантаження на веб-сайт.
  - Несправність маршрутизатора, що призводить до ізоляції частини мережі.

- Наслідки: Мережеві збої можуть унеможливити доступ до сервісів, спричинити затримки в передачі даних або перервати зв’язок між різними компонентами IT-інфраструктури. Наприклад, у фінансовій установі втрата зв’язку з сервером транзакцій може зупинити обробку платежів.
- Роль моніторингу: Мережевий моніторинг, реалізований через протоколи SNMP або ICMP, дозволяє відстежувати доступність хостів, пропускну здатність каналів, втрату пакетів і затримки. Системи, такі як Zabbix, можуть автоматично виявляти втрату зв’язку з вузлом (наприклад, якщо ping не повертає відповідь протягом 3 хвилин) і надсилати сповіщення адміністратору. Крім того, аналіз трендів мережевого трафіку допомагає планувати розширення пропускної здатності.

4. *Помилки програмного забезпечення.* Помилки в програмному забезпеченні можуть бути спричинені некоректною конфігурацією, багами в коді, несумісністю оновлень або неправильною інтеграцією між системами.

- Приклади помилок:
  - Збій веб-сервера Apache через неправильно налаштований конфігураційний файл.
  - Помилка в базі даних через несумісність після оновлення до нової версії.
  - Зависання корпоративного додатка через баг у коді, який викликає витік пам’яті.
- Наслідки: Помилки програмного забезпечення можуть призвести до зупинки сервісів, втрати даних або зниження продуктивності. Наприклад, збій у CRM-системі може перервати взаємодію з клієнтами, що негативно вплине на продажі.
- Роль моніторингу: Моніторинг додатків дозволяє відстежувати стан сервісів, час відгуку API, кількість помилок і продуктивність запитів. Наприклад, Zabbix може перевіряти статус служби (наприклад, чи запущений MySQL) і сповіщати про її зупинку. Крім того, системи моніторингу можуть аналізувати логи додатків для виявлення повторюваних помилок.

5. *Людський фактор.* Людський фактор є однією з основних причин IT-збоїв, оскільки помилки адміністраторів або користувачів можуть мати серйозні

наслідки. Це може включати неправильне налаштування систем, випадкове видалення даних або некоректне виконання процедур.

- Приклади людського фактора:
  - Випадкове видалення бази даних адміністратором під час виконання скрипта.
  - Некоректна конфігурація брандмауера, що блокує доступ до сервера.
  - Неправильне оновлення системи без попереднього тестування, що призводить до несумісності.
- Наслідки: Помилки, спричинені людським фактором, можуть мати катастрофічні наслідки, від втрати даних до повного припинення роботи системи. Наприклад, випадкове видалення критичної бази даних може зупинити бізнес-процеси на кілька годин або навіть днів.
- Роль моніторингу: Хоча моніторинг не може повністю усунути людський фактор, він може допомогти виявляти наслідки таких помилок. Наприклад, системи моніторингу можуть сповіщати про різке зменшення обсягу даних на диску, що може бути ознакою випадкового видалення. Крім того, автоматизація конфігураційних процесів через системи моніторингу зменшує ймовірність помилок.

6 *Атаки та шкідливе програмне забезпечення.* Кібератаки, такі як DDoS, віруси, ransomware або несанкціонований доступ, становлять серйозну загрозу для стабільності IT-систем.

- Приклади атак:
  - DDoS-атака, яка перевантажує веб-сервер запитами, роблячи його недоступним.
  - Вірус, що шифрує дані на сервері (ransomware), вимагаючи викуп.
  - Злом системи через вразливість у програмному забезпеченні, що дозволяє зловмисникам отримати доступ до конфіденційних даних.
- Наслідки: Кібератаки можуть призвести до втрати даних, простоїв, фінансових втрат і репутаційних ризиків. Наприклад, атака ransomware у медичній установі може заблокувати доступ до електронних медичних записів, що загрожує життю пацієнтів.

- Роль моніторингу: Системи моніторингу можуть виявляти аномалії, які вказують на кібератаки, наприклад, незвичайне зростання мережевого трафіку (можлива DDoS-атака) або підозрілі процеси, що споживають ресурси. Zabbix, наприклад, може налаштувати тригери для виявлення аномального використання CPU або мережі, а також інтегруватися з системами безпеки для аналізу логів.

7. Значення своєчасного виявлення проблем. Більшість перелічених проблем можна передбачити або мінімізувати їхній вплив, якщо вони виявлені на ранній стадії. Ефективна система моніторингу, така як Zabbix, відіграє ключову роль у цьому процесі, надаючи такі можливості:

- Проактивний підхід: Моніторинг дозволяє виявляти проблеми до їхнього переростання в критичні інциденти. Наприклад, поступове зростання використання диска може бути помічено за кілька тижнів до його повного заповнення.
- Автоматизація реагування: Системи моніторингу можуть автоматично виконувати дії, такі як перезапуск сервісу або надсилання сповіщень, що зменшує час простою.
- Аналіз трендів: Історичні дані допомагають виявляти закономірності, які вказують на потенційні проблеми, наприклад, періодичне перевантаження мережі в певний час дня.
- Зменшення впливу людського фактора: Автоматизовані тригери та сповіщення зменшують залежність від ручного втручання, що знижує ризик помилок.

1.3 Роль автоматизації у забезпеченні надійності IT-інфраструктури

Автоматизація є одним із ключових аспектів сучасного моніторингу IT-інфраструктури, що забезпечує її надійність, ефективність і масштабованість. У сучасних IT-середовищах, які характеризуються високою складністю, великими обсягами даних і динамічними змінами, ручний моніторинг стає не лише неефективним, але й практично неможливим. Без автоматизованих процесів адміністратори не в змозі вчасно реагувати на інциденти, обробляти численні

метрики або забезпечувати безперервний контроль за всіма компонентами системи. Автоматизація дозволяє оптимізувати ці процеси, зменшуючи час реакції на проблеми, мінімізуючи людський фактор і звільняючи ресурси IT-персоналу для виконання стратегічних завдань, таких як планування інфраструктури чи впровадження нових технологій.

Автоматизовані системи моніторингу, такі як Zabbix, Prometheus або Nagios, надають інструменти для збору, аналізу та реагування на дані в реальному часі. Вони дозволяють організаціям не лише фіксувати інциденти після їх виникнення, але й діяти проактивно, передбачаючи потенційні проблеми на основі трендів і аномалій. У цьому розділі детально розглянуто основні аспекти автоматизації в моніторингу IT-інфраструктури, її можливості, переваги, виклики та приклади практичного застосування.

Розглянемо основні можливості автоматизації. Автоматизація моніторингу охоплює широкий спектр функцій, які дозволяють ефективно управляти IT-інфраструктурою. Основні з них включають:

1. *Автоматичний збір метрик із великої кількості джерел.* Сучасні IT-системи генерують величезну кількість даних із різноманітних джерел: серверів, мережевих пристроїв, баз даних, хмарних сервісів, контейнерів тощо. Автоматизація дозволяє систематично й безперервно збирати ці метрики без необхідності ручного втручання. Наприклад, система Zabbix може використовувати протоколи SNMP, IPMI, WMI або API для періодичного опитування пристроїв і програм, забезпечуючи централізований збір даних про завантаженість процесора, використання пам’яті, мережевий трафік чи стан сервісів.

- Приклад: У корпоративній мережі з сотнями серверів і мережевих пристроїв Zabbix може автоматично отримувати метрики, такі як температура обладнання, пропускна здатність каналів або статус бази даних MySQL, кожні 30 секунд. Це усуває потребу в ручному моніторингу кожного пристрою.

2. Налаштування тригерів для реагування на відхилення
Тригери — це логічні умови, які визначають, коли показники системи відхиляються від норми. Автоматизація дозволяє створювати тригери, які

аналізують метрики в реальному часі та активують сповіщення або дії при виявленні аномалій. Наприклад, тригер може бути налаштований на виявлення ситуації, коли використання CPU перевищує 90% протягом 5 хвилин або коли дисковий простір зменшується до 10%.

- Приклад: У Zabbix тригер може бути налаштований так: {host:cpu.utilization.last()} > 90 and {host:cpu.utilization.time()} > 300. Якщо ця умова виконується, система автоматично позначає проблему як критичну й ініціює сповіщення.

3. *Генерація повідомлень про інциденти в реальному часі.* Автоматизовані системи моніторингу можуть миттєво інформувати адміністраторів про інциденти через різноманітні канали, такі як email, Telegram, SMS, Slack або webhook. Це значно скорочує час реакції на проблему, що є критично важливим для мінімізації простоїв.

- Приклад: У разі втрати зв’язку з сервером Zabbix може надіслати повідомлення в Telegram-бот із деталями, такими як ім’я хоста, час інциденту та його серйозність. Це дозволяє адміністратору негайно розпочати діагностику.

4. *Виконання скриптів або автоматичних дій.* Автоматизація дозволяє не лише виявляти проблеми, але й автоматично їх вирішувати шляхом виконання заздалегідь визначених скриптів або дій. Наприклад, система може перезапустити службу, очистити тимчасові файли або перенаправити мережевий трафік на резервний сервер.

- Приклад: Якщо Zabbix виявляє, що служба Apache зупинилася, система може автоматично виконати команду systemctl restart apache2 на відповідному сервері. Це зменшує час простою, особливо якщо інцидент відбувається поза робочим часом.

5. *Створення регулярних звітів і графіків змін.* Автоматизація забезпечує генерацію регулярних звітів про продуктивність системи, тренди використання ресурсів і статистику інцидентів. Графіки, дашборди та звіти створюються автоматично, надаючи адміністраторам і керівництву чітке уявлення про стан інфраструктури.

- Приклад: Zabbix може щотижня генерувати звіт, який показує середнє завантаження серверів, кількість інцидентів і час їх вирішення. Такі звіти допомагають планувати оновлення обладнання або оптимізацію ресурсів.

Розглянемо основні переваги автоматизації.

Автоматизація моніторингу приносить низку переваг, які суттєво підвищують надійність і ефективність IT-інфраструктури:

- Скорочення часу виявлення та вирішення проблем: Автоматизовані тригери та сповіщення дозволяють виявляти інциденти за лічені секунди, тоді як ручний моніторинг може займати години. Наприклад, якщо дисковий простір на сервері критично зменшується, автоматизована система сповістить адміністратора до того, як це призведе до зупинки сервісів.
- Покращення відновлення після інцидентів: Автоматичні дії, такі як перезапуск сервісів або перенаправлення трафіку, зменшують час простою. Це особливо важливо для організацій, де кожна хвилина простою коштує значних фінансових втрат.
- Зменшення впливу людського фактора: Автоматизація мінімізує залежність від ручного втручання, що знижує ризик помилок, таких як пропущені інциденти або неправильна конфігурація.
- Звільнення ресурсів IT-персоналу: Завдяки автоматизації рутинних завдань, таких як збір даних чи генерація звітів, адміністратори можуть зосередитися на стратегічних ініціативах, наприклад, впровадженні нових технологій або оптимізації архітектури.
- Масштабованість: Автоматизовані системи легко адаптуються до зростання інфраструктури. Наприклад, Zabbix підтримує автоматичне виявлення нових пристроїв у мережі (Low-Level Discovery), що дозволяє масштабувати моніторинг без додаткових зусиль.
- Прозорість і звітність: Автоматизовані звіти та дашборди забезпечують прозорість роботи IT-відділу, що є важливим для менеджменту та клієнтів. Це також допомагає дотримуватися угод про рівень обслуговування (SLA).

Розглянемо виклики автоматизації. Незважаючи на численні переваги, впровадження автоматизації в моніторингу пов’язане з певними викликами:

1. *Складність налаштування:* Налаштування тригерів, сповіщень і автоматичних дій вимагає ретельного планування. Неправильно налаштовані тригери можуть генерувати надмірну кількість сповіщень (alert fatigue) або пропускати критичні інциденти.

- *Рішення:* Використання шаблонів моніторингу, які надають системи, такі як Zabbix, дозволяє спростити конфігурацію та забезпечити стандартизацію.

2. *Ресурсоємність:* Автоматизовані системи можуть споживати значні обчислювальні ресурси, особливо в великих інфраструктурах із тисячами хостів.

- Рішення: Оптимізація системи, наприклад, використання проксі-серверів у Zabbix для розподіленого моніторингу, допомагає зменшити навантаження.

3. *Інтеграція з різними системами:* IT-інфраструктури часто включають різноманітні технології, які потребують інтеграції з моніторингом через різні протоколи та API.

- Рішення: Системи, такі як Zabbix, підтримують широкий спектр протоколів (SNMP, IPMI, HTTP, SSH), що полегшує інтеграцію.

4. *Безпека:* Автоматизовані дії, такі як виконання скриптів, можуть створювати ризики безпеки, якщо зловмисники отримають доступ до системи моніторингу.

- Рішення: Використання шифрування, авторизації та обмеження прав доступу в системах моніторингу забезпечує захист.

5. *Необхідність навчання персоналу:* Впровадження автоматизованих систем вимагає від IT-персоналу знань про їхню конфігурацію та управління.

- Рішення: Проведення тренінгів і використання офіційної документації, наприклад, Zabbix Documentation, допомагає подолати цей бар’єр.

Розглянемо практичні приклади автоматизації.

Автоматизація моніторингу широко застосовується в різних галузях. Ось кілька прикладів:

- Електронна комерція: Інтернет-магазин використовує Zabbix для моніторингу веб-сервера, бази даних і платіжного шлюзу. Якщо час відповіді веб-сайту перевищує 2 секунди, система автоматично надсилає сповіщення в Telegram і запускає скрипт для очищення кешу, що зменшує затримки.

- Фінансовий сектор: Банк застосовує автоматизований моніторинг для відстеження транзакційних серверів. Якщо кількість невдалих транзакцій перевищує 1% за хвилину, система перенаправляє трафік на резервний сервер і сповіщає адміністратора.
- Телекомунікації: Провайдер зв’язку використовує автоматизацію для моніторингу мережевих пристроїв. У разі втрати зв’язку з маршрутизатором система автоматично перемикає трафік на альтернативний маршрут і генерує звіт про інцидент.
- Виробництво: Виробниче підприємство моніторить IoT-пристрої на конвеєрі. Якщо датчик температури фіксує відхилення, система автоматично зупиняє конвеєр і надсилає сповіщення інженеру.

1.4 Структура, компоненти та критерії ефективності типових систем моніторингу

Більшість сучасних систем моніторингу мають подібну архітектуру:

- Агенти / SNMP / API — компоненти, що збирають дані з серверів, мережевого обладнання або застосунків;
- Сервер моніторингу — центральний елемент, що обробляє метрики, застосовує тригери та приймає рішення;
- База даних — для збереження історичних даних, логів, подій;
- Інтерфейс користувача — веб-панель для перегляду графіків, отримання алертів, налаштування параметрів;
- Система сповіщення — email, Telegram, SMS, Slack тощо;
- Зовнішні інтеграції — з CMDB, DevOps-інструментами, сервісними системами.

Ці елементи разом утворюють замкнуту систему, що дозволяє виявляти, аналізувати та реагувати на інциденти.

Перш ніж обрати конкретну систему моніторингу, необхідно визначити набір критеріїв, які дозволять об’єктивно оцінити ефективність кожного з можливих рішень. Такі критерії мають враховувати як технічні характеристики,

так і функціональні можливості систем, а також аспекти впровадження, адміністрування та масштабування.

Основні критерії ефективності систем моніторингу наведені нижче:

1. Підтримка різних джерел даних:
   - Чи підтримуються агентні та безагентні методи збору?
   - Чи є підтримка SNMP, IPMI, WMI, HTTP, SSH, API?
   - Чи можлива інтеграція з хмарними сервісами (AWS, Azure, GCP)?
2. Гнучкість налаштувань:
   - Наскільки гнучко можна створювати власні шаблони моніторингу?
   - Чи можна визначати власні метрики?
   - Чи передбачена система тригерів та алертів з розширеною логікою?
3. Масштабованість:
   - Чи може система працювати в великих середовищах (1000+ хостів)?
   - Як працює зберігання історичних даних?
   - Чи є механізми розподіленого моніторингу?
4. Інтерфейс користувача:
   - Наявність графічного веб-інтерфейсу, зручного для аналізу.
   - Побудова дашбордів та графіків.
   - Наявність мобільної підтримки.
5. Механізми сповіщень:
   - Чи є підтримка e-mail, SMS, Telegram, Slack, Webhook тощо?
   - Чи можна задавати умови для надсилання сповіщень?
   - Чи є підтримка ескалації?
6. Продуктивність та вимоги до ресурсів:
   - Скільки ресурсів споживає система на 100, 1000 хостах?
   - Чи є потреба в окремих серверах БД, проксі?
7. Безпека:
   - Чи підтримується шифрування переданих даних?
   - Чи можлива інтеграція з LDAP, Active Directory?
   - Чи є система ролей та прав доступу?
8. Ліцензування і вартість:

- Відкритий код чи пропрієтарне ПЗ?
- Чи потребує комерційної ліцензії?
- Які обмеження на безкоштовне використання?

9. Документація та спільнота:
- Чи є повна офіційна документація?
- Наскільки активна спільнота?
- Наявність готових шаблонів, скриптів, форумів.

10. Можливість інтеграції:
- Чи можна підключити до зовнішніх CMDB, ITSM систем?
- Підтримка API для інтеграції з DevOps CI/CD?
- Можливість імпорту даних з інших систем моніторингу?

Зважаючи на ці критерії, у подальших підрозділах буде проведено порівняння чотирьох популярних систем моніторингу: Zabbix, Nagios, Prometheus та Icinga. Кожна з них буде розглянута окремо, а згодом буде представлена порівняльна таблиця, яка дозволить обґрунтовано зробити вибір найкращого варіанту для реалізації поставленої задачі.

1.5 Огляд сучасних систем моніторингу

На сьогоднішній день на ринку представлено велику кількість систем моніторингу. Їх можна класифікувати за різними ознаками:

- За моделлю розгортання:
  - Локальні (on-premises): Zabbix, Nagios, Icinga;
  - Хмарні: Datadog, New Relic, Dynatrace.
- За ліцензуванням:
  - Відкриті (open-source): Zabbix, Prometheus, Grafana (у парі з іншими);
  - Комерційні: Paessler PRTG, SolarWinds, Splunk.
- За функціональністю:
  - Орієнтовані на інфраструктуру: Zabbix, Nagios;
  - Для додатків і DevOps: Prometheus, Dynatrace;
  - Гібридні: Datadog, Icinga.

Кожна з систем має свої переваги і недоліки та надає широкі можливості автоматизації [15]. Наприклад, Zabbix забезпечує повноцінний контроль за інфраструктурою, але вимагає конфігурації. Prometheus добре інтегрується з Kubernetes, але не має вбудованої бази даних історичних метрик. Datadog має інтуїтивний інтерфейс, але є платним.

Розглянемо найпопулярніші системи-аналоги більш детально.

1. **Zabbix** — це повнофункціональна система моніторингу з відкритим вихідним кодом, яка дозволяє здійснювати збір, зберігання, аналіз та візуалізацію даних у реальному часі[2]. Вона підтримує як агентний, так і безагентний моніторинг, працює з широким спектром протоколів (SNMP, IPMI, HTTP, SSH, Telnet, JMX, VMware API) та має потужну систему сповіщень. Однією з ключових переваг є розширюваність через шаблони, можливість масштабування за допомогою проксі-серверів та централізоване управління великою кількістю хостів.

Система надає гнучкі механізми створення тригерів з логічними умовами, історичне зберігання даних у СУБД (MySQL, PostgreSQL тощо), створення інтерактивних дашбордів та інтеграцію зі сторонніми сервісами. Zabbix активно підтримується спільнотою та має велику кількість готових шаблонів.

2. Nagios — одна з найстаріших систем моніторингу, яка стала основою для багатьох інших проєктів[3]. Основна концепція — це моніторинг за допомогою плагінів, які виконують перевірки стану сервісів або ресурсів. Nagios підтримує перевірку як локальних, так і віддалених хостів, має систему сповіщень та логічних умов (alerting), але порівняно обмежений у плані масштабування та візуалізації.

Варто зазначити, що базовий функціонал Nagios є досить мінімалістичним, і для забезпечення сучасного рівня зручності потрібно підключати сторонні надбудови (наприклад, графіки через PNP4Nagios або Thruk). При цьому Nagios вирізняється стабільністю та простотою архітектури, але вимагає більше ручного налаштування.

3. Prometheus - система моніторингу та збору метрик, розроблена в рамках проєкту Cloud Native Computing Foundation (CNCF)[4]. Вона орієнтована на збір

числових метрик у форматі time series (серії часу) і є ідеальним рішенням для моніторингу контейнеризованих середовищ (Kubernetes, Docker).

4. **Icinga** — це форк Nagios, створений з метою модернізації та розширення функціональності[5]. Система має оновлений веб-інтерфейс, REST API, потужні можливості інтеграції та підтримку сучасних технологій. Однією з ключових переваг є модульна архітектура: Icinga 2 виконує перевірки, а Icinga Web 2 забезпечує візуалізацію та адміністрування.

Icinga підтримує всі плагіни від Nagios, але при цьому додає власні механізми масштабування, автоматичного розгортання конфігурацій та взаємодії з базами даних. Це робить її більш гнучкою для великих середовищ, порівняно з класичним Nagios. Незважаючи на складнішу інсталяцію, система широко використовується у середовищах, де потрібна стабільність та розширюваність.

Порівняння розглянутих систем моніторингу наведено у таблиці 1.1

Таблиця 1.1 - Порівняння систем моніторингу

Критерій	Zabbix	Nagios	Prometheus	Icinga
Тип моніторингу	Агентний/ безагентний	Агентний	Безагентний	Агентний/ безагентний
Модель збору даних	Push/Pull	Push	Pull	Push
Масштабованість	Висока	Середня	Висока	Висока
Візуалізація	Вбудована	Через сторонні засоби	Через Grafana	Вбудована (Web 2)
Гнучкість конфігурації	Висока	Обмежена	Висока	Висока
Система сповіщень	Потужна, гнучка	Базова	Alertmanager	Потужна
Складність налаштування	Середня	Висока	Середня	Висока
Інтеграції та API	REST API, SNMP, SSH	Обмежені	REST, gRPC	REST API
Підтримка кластеризації	Так (через проксі)	Ні	Так (Thanos, Cortex)	Так
Підтримка/ актуальність	Активна, регулярні оновлення	Стабільна, але застаріла	Дуже активна (CNCF)	Активна, сучасна

1.6 Висновки до розділу 1

Моніторинг IT-інфраструктури є критично важливим процесом для забезпечення стабільності, безпеки та ефективності сучасних організацій. Він охоплює кілька рівнів — від інфраструктурного до користувацького досвіду — і виконує такі завдання, як раннє виявлення проблем, превентивне реагування та оптимізація ресурсів. У контексті ITSM моніторинг забезпечує прозорість, відповідність SLA та покращення якості IT-послуг.

Сучасні виклики, такі як великі обсяги даних, динамічність середовищ і різноманітність технологій, роблять автоматизацію моніторингу необхідною умовою. Системи, такі як Zabbix, Nagios і Prometheus, пропонують потужні інструменти для автоматизованого збору даних, аналізу та реагування на інциденти, що дозволяє організаціям ефективно управляти своєю IT-інфраструктурою. У наступних розділах буде детально розглянуто особливості таких систем, їх порівняння та практичне застосування.

2 ДОСЛІДЖЕННЯ СИСТЕМИ МОНІТОРИНГУ З ВИКОРИСТАННЯМ ZABBIX

2.1 Дослідження можливих напрямків покращення систем моніторингу з використанням сучасних інформаційних систем

Проаналізуємо перспективи розвитку систем моніторингу з точки зору AI, хмар та аналітики, що забезпечують сучасні інформаційні системи.

Зі швидким розвитком інформаційних технологій і зростанням складності IT-інфраструктур системи моніторингу зазнають значних змін. Сучасні IT-середовища характеризуються розподіленими архітектурами, хмарними платформами, мікросервісами та величезними обсягами даних, що генеруються в реальному часі. Для ефективного управління такими системами традиційні підходи до моніторингу, які базуються на статичних правилах і ручному аналізі, стають недостатніми. Нові технології, такі як штучний інтелект (AI), машинне навчання, хмарні обчислення та аналітика великих даних, відкривають нові можливості для створення більш інтелектуальних, адаптивних і масштабованих систем моніторингу.

Цей розділ детально розглядає ключові напрямки розвитку систем моніторингу, включаючи використання AI для прогнозування та аналізу, адаптацію до хмарних і гібридних середовищ, а також інтеграцію з аналітикою великих даних для обробки метрик і побудови звітів. Окремо аналізуються приклади реалізації, виклики впровадження та потенційні перспективи для майбутнього розвитку.

1. Штучний інтелект і машинне навчання. Штучний інтелект (AI) і машинне навчання (ML) революціонізують підходи до моніторингу IT-інфраструктури, дозволяючи системам не лише реагувати на проблеми, але й передбачати їх і автоматично знаходити першопричини. AI-модулі значно підвищують точність і ефективність моніторингу завдяки здатності обробляти складні набори даних і виявляти закономірності, які важко помітити традиційними методами. Основні напрямки використання AI/ML у моніторингу включають:

- *Anomaly Detection (Виявлення аномалій).* Алгоритми машинного навчання можуть аналізувати історичні дані та поведінку системи, щоб виявляти відхилення від нормального стану без необхідності задавати статичні порогові значення. Наприклад, якщо веб-сервер зазвичай обробляє 1000 запитів за хвилину, але раптово їхня кількість зростає до 5000, AI може позначити це як аномалію, навіть якщо порогове значення не було порушено.

  - Приклад: Система Dynatrace використовує свій AI-движок Davis для аналізу метрик у реальному часі, виявляючи аномалії, такі як незвичайне зростання часу відгуку API, і автоматично сповіщаючи адміністраторів.

- *Predictive Maintenance (Прогнозне обслуговування).* AI дозволяє прогнозувати потенційні збої обладнання або перевантаження системи на основі історичних трендів і патернів. Наприклад, аналізуючи дані про температуру жорстких дисків і частоту помилок SMART, система може передбачити їх вихід із ладу за кілька тижнів до поломки.

  - Приклад: У центрі обробки даних AI-модуль може виявити, що сервер регулярно перегрівається в певний час доби через пікове навантаження, і запропонувати оптимізувати систему охолодження або перерозподілити навантаження.

- *Root Cause Analysis (RCA, Аналіз першопричин).* AI здатен автоматично визначати першопричини інцидентів, аналізуючи взаємозв’язки між метриками різних компонентів системи. Наприклад, якщо веб-сайт стає недоступним, AI може виявити, що проблема пов’язана з перевантаженням бази даних, а не з веб-сервером.

  - Приклад: Splunk використовує машинне навчання для кореляції подій, що дозволяє швидко визначити, чи збій у роботі додатка спричинений мережевою затримкою, помилкою в коді чи апаратною несправністю.

- *Прогнозування навантаження.* AI може аналізувати історичні дані про використання ресурсів, щоб прогнозувати пікові навантаження та рекомендувати масштабування інфраструктури. Наприклад, у період розпродажів інтернет-магазин може заздалегідь збільшити кількість серверів на основі прогнозів AI.

- Приклад: AWS CloudWatch із функцією Anomaly Detection може прогнозувати зростання трафіку для веб-додатків і автоматично запускати додаткові екземпляри серверів через Auto Scaling.

Системи моніторингу, такі як Zabbix, уже починають інтегрувати AI-функціонал через плагіни або зовнішні модулі. Наприклад, Zabbix може використовувати API для зв'язку з платформами ML, такими як TensorFlow або Python-скрипти, для аналізу метрик і прогнозування. У майбутньому очікується глибша інтеграція AI в ядро таких систем, що зробить їх більш автономними та інтелектуальними.

Виклики впровадження AI

- Якість даних: AI потребує великих обсягів чистих і структурованих даних для навчання моделей. Некоректні або неповні метрики можуть призвести до хибних прогнозів.
- Складність впровадження: Налаштування AI-моделей вимагає спеціалізованих знань і може бути дорогим для невеликих організацій.
- Інтерпретація результатів: AI може виявляти аномалії, але пояснення їх причин може бути складним для адміністраторів без досвіду в ML.

Рішення

- Використання готових AI-платформ, таких як Dynatrace або Splunk, які пропонують попередньо навчені моделі.
- Інтеграція з відкритими бібліотеками ML (наприклад, scikit-learn) для кастомізації.
- Проведення регулярного навчання персоналу для роботи з AI-інструментами.

2. Хмарні та гібридні середовища. Зі зростанням популярності хмарних технологій, таких як Amazon Web Services (AWS), Microsoft Azure і Google Cloud Platform (GCP), а також гібридних інфраструктур, що поєднують локальні та хмарні ресурси, системи моніторингу повинні адаптуватися до динамічних і розподілених середовищ. Хмарні платформи характеризуються високою мінливістю, де ресурси створюються, знищуються або масштабуються в реальному часі, що створює нові виклики для моніторингу.

- *Моніторинг Kubernetes-кластерів.* Kubernetes, як провідна платформа для оркестрації контейнерів, вимагає спеціалізованих інструментів моніторингу для відстеження подів, нод, контейнерів і сервісів. Системи моніторингу повинні підтримувати автоматичне виявлення нових ресурсів (Service Discovery) і обробку метрик, таких як використання CPU/пам'яті подами або кількість реплік сервісу.

  - Приклад: Prometheus із інтеграцією Kubernetes Service Discovery автоматично виявляє нові поди й збирає метрики через kube-state-metrics. Zabbix також підтримує моніторинг Kubernetes через шаблони та API.

- *Автоматичне масштабування на основі метрик.* Хмарні платформи дозволяють автоматично масштабувати ресурси на основі метрик, таких як завантаженість серверів або кількість запитів. Системи моніторингу відіграють ключову роль у наданні даних для таких рішень.

  - Приклад: AWS Auto Scaling може використовувати метрики від Zabbix або CloudWatch, щоб запускати додаткові екземпляри EC2, коли кількість запитів до веб-сервера перевищує певний поріг.

- *Моніторинг мікросервісів у хмарі.* Мікросервісні архітектури, які широко використовуються в хмарних середовищах, потребують моніторингу кожної служби окремо, а також їхньої взаємодії. Це включає відстеження API-ендпоінтів, затримок між сервісами та помилок.

  - Приклад: Zabbix може використовувати HTTP-запити для перевірки доступності мікросервісів, а Grafana — для створення дашбордів, що відображають залежності між сервісами.

Системи моніторингу, такі як Zabbix, уже пропонують шаблони для AWS, Azure і GCP, що дозволяють відстежувати метрики хмарних ресурсів, таких як EC2, RDS або Google Compute Engine. Проте для повноцінної підтримки cloud-native екосистем потрібна глибша інтеграція з інструментами, такими як Prometheus, Grafana Loki або Elastic Observability, які спеціально розроблені для хмарних середовищ.

Виклики моніторингу хмар:

- Динамічність ресурсів: Хмарні ресурси швидко змінюються, що ускладнює їх відстеження.

- Вартість: Моніторинг великої кількості хмарних ресурсів може бути дорогим через плату за API-запити.
- Складність інтеграції: Різні хмарні провайдери мають власні API та формати метрик.

Рішення викликів моніторингу хмар:

- Використання шаблонів Service Discovery у Zabbix для автоматичного виявлення нових ресурсів.
- Інтеграція з хмарними платформами через API та webhook.
- Оптимізація збору метрик для зменшення витрат, наприклад, вибір лише критичних метрик.

3. **Big Data та аналітика.** Зростання обсягів даних, які генеруються IT-системами, вимагає нових підходів до їх обробки, зберігання та аналізу. Сучасні системи моніторингу повинні не лише збирати метрики в реальному часі, але й забезпечувати їх довгострокове зберігання, статистичний аналіз і інтеграцію з інструментами бізнес-аналітики (BI). Аналітика великих даних відкриває можливості для створення детальних звітів, прогнозування трендів і оцінки відповідності угодам про рівень обслуговування (SLA/SLI).

- *Статистичний аналіз історичних метрик.* Історичні дані дозволяють виявляти довгострокові тренди, такі як поступове зростання використання ресурсів або сезонні піки навантаження. Це допомагає планувати оновлення інфраструктури або оптимізувати її.
  - Приклад: Zabbix може зберігати історичні дані про використання CPU за рік, дозволяючи виявити, що навантаження зростає щокварталу через звітний період компанії.
- *Зв'язок із BI-системами.* Інтеграція систем моніторингу з BI-платформами, такими як Grafana, Power BI або Tableau, дозволяє створювати інтерактивні дашборди та звіти, які зрозумілі не лише технічним спеціалістам, але й керівництву.
  - Приклад: Grafana, інтегрована з Zabbix через плагін, може створювати дашборд, який відображає доступність сервісів, кількість інцидентів і продуктивність системи в реальному часі.

- *Побудова SLA/SLI звітів.* Аналітика великих даних дозволяє оцінювати відповідність інфраструктури угодам про рівень обслуговування (SLA) через метрики, такі як час безвідмовної роботи (uptime) або середній час відновлення після збою (MTTR).
  - Приклад: Zabbix може генерувати звіт SLA, який показує, що сервер був доступний 99.9% часу за місяць, що відповідає вимогам клієнта.

Для обробки великих обсягів метрик системи моніторингу інтегруються з платформами, такими як Elasticsearch, InfluxDB або ClickHouse, які спеціалізуються на зберіганні та аналізі часових рядів. Наприклад, InfluxDB може зберігати мільйони метрик із високою швидкістю запису, а Elasticsearch — забезпечувати повнотекстовий пошук по логах для аналізу інцидентів.

Виклики аналітики великих даних

- Обсяг даних: Зберігання й обробка великих обсягів метрик потребує значних обчислювальних ресурсів.
- Складність аналізу: Інтерпретація великих наборів даних вимагає спеціалізованих знань і інструментів.
- Інтеграція: Поєднання систем моніторингу з BI-платформами може бути складним через різні формати даних.

Рішення проблем аналітики великих даних:

- Використання спеціалізованих баз даних для часових рядів, таких як InfluxDB або TimescaleDB.
- Інтеграція через API та плагіни, наприклад, Zabbix-Grafana connector.
- Автоматизація створення звітів за допомогою скриптів або вбудованих інструментів.

4. **Інтеграція з DevOps і CI/CD.** Сучасні системи моніторингу все частіше інтегруються з практиками DevOps, щоб забезпечити моніторинг не лише на етапі експлуатації, але й під час розробки та розгортання додатків. Це включає:

- Моніторинг CI/CD-пайплайнів: Відстеження продуктивності процесів безперервної інтеграції та доставки.
- Observability: Поєднання моніторингу метрик, логів і трасування для повного розуміння поведінки системи.

- Інтеграція з інструментами DevOps: Наприклад, Prometheus і Grafana широко використовуються в DevOps для моніторингу мікросервісів, а Zabbix може бути адаптований через API.
  - Приклад: У DevOps-команді Zabbix може відстежувати продуктивність Jenkins-пайплайну, сповіщаючи про затримки в розгортанні нових версій додатка.

5. **Майбутнє систем моніторингу.** У майбутньому системи моніторингу стануть ще більш інтелектуальними та інтегрованими. Основні тенденції включають:

- Автономний моніторинг: AI-системи зможуть самостійно налаштовувати тригери, оптимізувати метрики та виконувати дії без участі адміністратора.
- Edge Computing: Моніторинг периферійних пристроїв (IoT, edge-сервери) потребуватиме нових підходів до збору й обробки даних у реальному часі.
- Уніфікована Observability: Поєднання метрик, логів, трасування та подій в єдиній платформі для повного аналізу системи.
- Безпека: Системи моніторингу будуть інтегруватися з інструментами кібербезпеки для виявлення атак у реальному часі.

2.2 Постановка задачі: автоматизоване виявлення проблем

Забезпечення безперебійної роботи IT-інфраструктури є однією з основних умов функціонування сучасних організацій, незалежно від їхньої сфери діяльності — від електронної комерції до фінансових установ чи промислових підприємств. IT-інфраструктура, що включає сервери, мережеві пристрої, хмарні сервіси, бази даних і додатки, є основою для бізнес-процесів, таких як обробка транзакцій, управління клієнтськими даними чи автоматизація виробництва. Однак зі зростанням складності та масштабів цифрових сервісів кількість потенційних точок відмови значно збільшується, що створює серйозні виклики для забезпечення продуктивності, доступності та безпеки IT-систем.

Традиційні підходи до моніторингу, які базуються на ручному аналізі, періодичних перевірках або звітах від кінцевих користувачів, є неефективними в сучасних динамічних IT-середовищах. Такі методи характеризуються повільною

реакцією, суб’єктивністю та високою ймовірністю пропуску критичних інцидентів. Наприклад, користувач може повідомити про проблему з веб-додатком лише після того, як вона вже спричинила значні незручності, а адміністратор може не помітити поступового погіршення продуктивності сервера через відсутність автоматизованого аналізу. У результаті інциденти часто виявляються із запізненням, коли їхній негативний вплив на бізнес-процеси вже є значним.

У цьому контексті автоматизоване виявлення проблем набуває критичного значення. Воно дозволяє не лише оперативно реагувати на інциденти, але й діяти проактивно, передбачаючи потенційні збої на основі трендів і аномалій. Метою цього розділу є чітка постановка задачі створення автоматизованої системи моніторингу, яка забезпечує ефективне виявлення проблем на всіх рівнях IT-інфраструктури, адаптується до специфічних вимог середовища та підтримує історичний аналіз для прогнозування.

Проблема традиційних методів моніторингу. Традиційні методи моніторингу мають низку обмежень, які роблять їх непридатними для сучасних IT-систем:

- Повільність реагування: Ручний моніторинг залежить від періодичних перевірок адміністратором, що може займати години або навіть дні, особливо в складних інфраструктурах із сотнями пристроїв.
- Суб’єктивність: Виявлення проблем часто залежить від досвіду адміністратора або скарг користувачів, що може призводити до пропуску прихованих або поступових проблем, таких як повільне зростання використання ресурсів.
- Обмежена масштабованість: У великих системах, що включають хмарні платформи, контейнери чи мікросервіси, ручне відстеження всіх компонентів є неможливим.
- Відсутність проактивності: Традиційні методи зазвичай фіксують інциденти після їх виникнення, не дозволяючи передбачити проблеми на основі трендів.

Ці недоліки підкреслюють необхідність автоматизації, яка забезпечує безперервний моніторинг, швидке виявлення проблем і автоматичне реагування, мінімізуючи вплив інцидентів на бізнес-процеси.

Ключові завдання автоматизованої системи моніторингу. Основною задачею є розробка та впровадження автоматизованої системи моніторингу, яка забезпечує проактивне виявлення проблем у реальному часі. Така система повинна виконувати наступні функції:

1. *Автоматичний збір даних із різноманітних джерел.* Система має забезпечувати централізований збір метрик із серверів, мережевих пристроїв, хмарних сервісів, баз даних і додатків. Для цього необхідно підтримувати різні протоколи (SNMP, IPMI, WMI, HTTP, API) і методи збору даних (агентний і безагентний моніторинг). Наприклад, Zabbix може використовувати SNMP для збору даних із маршрутизаторів Cisco, API для моніторингу хмарних сервісів AWS і агенти для відстеження стану серверів Linux.

- Приклад: У корпоративній мережі система автоматично збирає метрики про завантаженість CPU серверів, пропускну здатність мережевих інтерфейсів і час відгуку бази даних MySQL кожні 30 секунд.

2. *Аналіз даних у реальному часі.* Система повинна обробляти зібрані метрики в реальному часі, порівнюючи їх із нормальними значеннями або історичними трендами. Це включає виявлення аномалій, таких як різке зростання мережевого трафіку чи зниження продуктивності додатка.

- Приклад: Якщо час відгуку API веб-додатка перевищує 2 секунди, система автоматично позначає це як потенційну проблему й запускає подальший аналіз.

3. Виявлення відхилень, збоїв і погіршення продуктивності Автоматизована система має використовувати тригери та логічні умови для виявлення відхилень від нормального стану. Наприклад, тригер може активуватися, якщо вільний дисковий простір падає нижче 10% або якщо кількість невдалих запитів до бази даних перевищує 5% за хвилину.

- Приклад: Zabbix може налаштувати тригер {host:free.disk.space.perc.last()} < 10, який сповіщає адміністратора про критичне зменшення дискового простору.

4. *Формування повідомлень відповідальним особам*. Система має автоматично надсилати сповіщення через різні канали (Telegram, email, SMS, Slack) із деталями про інцидент, включаючи його серйозність, час виникнення та уражений компонент. Це дозволяє адміністраторам швидко реагувати на проблему.

- Приклад: У разі зупинки служби Apache система надсилає повідомлення в Telegram-бот із текстом: “Критичний інцидент: служба Apache зупинена на сервері web01 о 14:23”.

5. *Зберігання інформації для подальшого аналізу*. Усі зібрані метрики та журнали подій повинні зберігатися в базі даних для історичного аналізу, прогнозування трендів і створення звітів. Це дозволяє виявляти закономірності, такі як періодичне зростання навантаження, і планувати відповідні дії.

- Приклад: Zabbix зберігає дані про використання CPU за рік, що дозволяє виявити, що навантаження зростає щомісяця через регулярні оновлення бази даних.

Розглянемо рівні моніторингу. Автоматизоване виявлення проблем має охоплювати всі основні рівні IT-інфраструктури, щоб забезпечити комплексний контроль:

- *Мережевий рівень:* Моніторинг доступності вузлів (ping, SNMP), пропускної здатності каналів, втрати пакетів і затримок (latency). Наприклад, система може виявити втрату зв’язку з маршрутизатором або перевантаження мережевого каналу через надмірний трафік.

- Приклад: Тригер у Zabbix активується, якщо втрата пакетів на мережевому інтерфейсі перевищує 2%, що може вказувати на проблему з провайдером.

- *Серверний рівень:* Відстеження використання ресурсів (CPU, RAM, диск), стану операційних систем (Windows, Linux) і служб (наприклад, SSH, Nginx). Система має виявляти такі проблеми, як перевантаження процесора або зупинка критичної служби.

- Приклад: Якщо використання RAM на сервері перевищує 95% протягом 5 хвилин, Zabbix надсилає сповіщення та запускає скрипт для очищення кешу.

- *Прикладний рівень:* Моніторинг працездатності веб-сервісів, API, баз даних і корпоративних додатків. Це включає перевірку часу відгуку, кількості помилок і доступності сервісів.

- Приклад: Zabbix може періодично надсилати HTTP-запити до веб-сайту й активувати тригер, якщо статус-код відповіді дорівнює 503 (Service Unavailable).

Адаптація до особливостей середовища. Автоматизована система моніторингу має бути гнучкою, щоб адаптуватися до специфіки конкретного IT-середовища. Основні вимоги включають:

- Підтримка різних протоколів: Система повинна працювати з SNMP, IPMI, WMI, SSH, HTTP і API для збору даних із різноманітних пристроїв і сервісів. Наприклад, SNMP підходить для мережевого обладнання, а API — для хмарних платформ, таких як AWS.

- Інтеграція з хмарними інфраструктурами: Система має підтримувати моніторинг хмарних ресурсів (EC2, RDS, Azure VMs) і динамічних середовищ, таких як Kubernetes-кластери, де ресурси постійно змінюються.

- Приклад: Zabbix використовує шаблони для AWS, які дозволяють відстежувати метрики EC2, такі як використання CPU або обсяг мережевого трафіку.

- Робота в умовах обмежених ресурсів: У невеликих організаціях із обмеженим бюджетом система має бути ефективною, мінімізуючи споживання обчислювальних ресурсів.

- Приклад: Zabbix може використовувати проксі-сервери для розподіленого моніторингу, зменшуючи навантаження на основний сервер.

- Гнучке налаштування тригерів: Система повинна дозволяти створювати кастомні тригери, які враховують специфіку інфраструктури, наприклад, різні порогові значення для серверів із різним навантаженням.

- Приклад: Для сервера бази даних тригер може бути налаштований на 80% використання CPU, тоді як для веб-сервера — на 90%.

Історичний аналіз і прогнозування. Окремим важливим аспектом є історичний аналіз і прогнозування, які дозволяють не лише реагувати на поточні проблеми, але й передбачати майбутні. Це включає:

- Аналіз трендів: Виявлення закономірностей, таких як поступове зростання використання диска чи періодичні піки мережевого трафіку.
  - Приклад: Аналіз історичних даних у Zabbix може показати, що використання пам’яті зростає на 5% щомісяця, що вказує на потребу в додаткових ресурсах.
- Прогнозування збоїв: Використання статистичних методів або алгоритмів машинного навчання для передбачення потенційних проблем, таких як вихід із ладу жорсткого диска.
  - Приклад: На основі даних SMART система може спрогнозувати поломку диска за два тижні до її виникнення.
- Побудова звітів: Генерація звітів про продуктивність, інциденти та відповідність SLA для оцінки ефективності інфраструктури.
  - Приклад: Zabbix може створювати щомісячний звіт, який показує середній час безвідмовної роботи (uptime) і кількість критичних інцидентів.

Практичні вимоги до системи

Для реалізації автоматизованої системи моніторингу необхідно врахувати наступні практичні аспекти:

- Масштабованість: Система має підтримувати моніторинг сотень або тисяч пристроїв без втрати продуктивності. Наприклад, Zabbix може використовувати розподілену архітектуру з проксі-серверами для великих мереж.
- Інтеграція зі сповіщеннями: Система повинна підтримувати популярні канали зв’язку, такі як Telegram, Slack або email, для оперативного інформування адміністраторів.
- Безпека: Доступ до системи моніторингу має бути захищений авторизацією, шифруванням і контролем прав, щоб запобігти несанкціонованому доступу.
  - Приклад: Zabbix підтримує шифрування з’єднань між агентами та сервером за допомогою TLS.

- Інтерфейс користувача: Система має надавати зручний веб-інтерфейс із дашбордами для візуалізації метрик і стану інфраструктури.
  - Приклад: Zabbix пропонує кастомізовані дашборди, які відображають графіки використання ресурсів і статус тригерів.

Виклики реалізації. Впровадження автоматизованої системи моніторингу пов'язане з низкою викликів:

- Складність налаштування: Визначення правильних тригерів і порогових значень вимагає глибокого розуміння інфраструктури, щоб уникнути надмірних сповіщень (alert fatigue) або пропуску критичних інцидентів.
- Ресурсоємність: Збір і обробка великих обсягів метрик можуть перевантажувати сервер моніторингу, особливо в масштабних системах.
- Інтеграція з різними платформами: Різноманітність обладнання та програмного забезпечення (локальні сервери, хмарні платформи, контейнери) ускладнює уніфікацію моніторингу.
- Навчання персоналу: Адміністратори повинні мати навички роботи з системами моніторингу та розуміння специфіки протоколів і API.

Рішення викликів. Для подолання цих викликів пропонується:

- Використання шаблонів моніторингу, таких як ті, що надаються Zabbix, для спрощення налаштування.
- Застосування розподілених архітектур (наприклад, Zabbix Proxy) для зменшення навантаження.
- Інтеграція через API для підтримки хмарних і гібридних середовищ.
- Проведення тренінгів для IT-персоналу та використання документації, наприклад, Zabbix Documentation.

2.3 Загальна характеристика системи Zabbix

Zabbix — це повноцінна платформа моніторингу з відкритим вихідним кодом, яка підтримує моніторинг мереж, серверів, додатків і сервісів. Архітектура Zabbix є модульною та розподіленою, що забезпечує її гнучкість і масштабованість. Основні компоненти системи включають:

- Zabbix Server: Центральний компонент, який відповідає за обробку даних, виконання тригерів, генерацію сповіщень і управління конфігурацією. Сервер взаємодіє з базою даних і клієнтськими агентами.
- Zabbix Agent: Легкий агент, що встановлюється на пристрої для збору метрик (наприклад, CPU, пам’ять, дисковий простір). Підтримує як активний (ініціюється агентом), так і пасивний (ініціюється сервером) режими моніторингу.
- Zabbix Proxy: Додатковий компонент для розподіленого моніторингу, який збирає дані від пристроїв у віддалених мережах і передає їх на центральний сервер. Це зменшує навантаження на сервер і забезпечує моніторинг у мережах із обмеженим доступом.
- База даних: Зберігає конфігурацію, зібрані метрики та історію подій. Zabbix підтримує популярні СУБД, такі як MySQL, PostgreSQL, Oracle і TimescaleDB.
- Веб-інтерфейс: Інтуїтивно зрозумілий інтерфейс для налаштування, візуалізації даних (графіки, дашборди, карти) і управління системою.
- API: Дозволяє інтегрувати Zabbix із зовнішніми системами, такими як системи управління інцидентами, BI-платформи чи хмарні сервіси.

Така архітектура дозволяє Zabbix адаптуватися до різноманітних IT-середовищ, від локальних серверів до гібридних і хмарних інфраструктур.

Основні характеристики Zabbix

Zabbix пропонує широкий набір функцій, які роблять його універсальним інструментом для моніторингу. Основні характеристики:

1. *Підтримка активного та пасивного моніторингу.* Zabbix підтримує два режими збору даних:

- Активний моніторинг: Агент самостійно надсилає метрики на сервер за заданим графіком, що зменшує навантаження на мережу.
- Пасивний моніторинг: Сервер періодично опитує пристрої, що корисно для обладнання без встановлених агентів.
- Приклад: Для серверів Linux Zabbix Agent може активно надсилати дані про використання CPU кожні 30 секунд, тоді як для мережевого комутатора Cisco використовується пасивний моніторинг через SNMP.

2. *Гнучка система тригерів і шаблонів.* Zabbix дозволяє створювати складні тригери, які реагують на відхилення метрик від заданих порогів. Тригери базуються на логічних виразах, таких як {host:cpu.utilization.last()} > 90. Шаблони спрощують налаштування, дозволяючи застосовувати однакові параметри моніторингу до груп пристроїв.

- Приклад: Шаблон “Linux Servers” у Zabbix автоматично застосовує тригери для моніторингу CPU, RAM і дискового простору до всіх серверів із ОС Linux.

3. *Візуалізація даних.* Zabbix надає потужні інструменти для візуалізації у вигляді графіків, мережевих карт, дашбордів і теплових карт. Користувачі можуть створювати кастомізовані дашборди, які відображають ключові метрики в реальному часі.

- Приклад: Дашборд у Zabbix може показувати графіки використання мережі, статус серверів і кількість активних тригерів для дата-центру.

4. *Масштабованість.* Zabbix підходить як для невеликих мереж із кількома пристроями, так і для великих інфраструктур із тисячами хостів. Використання проксі-серверів і оптимізованих баз даних, таких як TimescaleDB, дозволяє масштабувати систему без втрати продуктивності.

- Приклад: Телекомунікаційна компанія може використовувати Zabbix для моніторингу 10,000 мережевих пристроїв, розподілених по різних регіонах, із проксі-серверами в кожному офісі.

5. *Підтримка різноманітних протоколів.* Zabbix підтримує численні протоколи та методи збору даних, включаючи SNMP (Simple Network Management Protocol), IPMI (Intelligent Platform Management Interface), JMX (Java Management Extensions), SSH, Telnet, HTTP-запити та API. Це дозволяє моніторити широкий спектр пристроїв і сервісів, від маршрутизаторів до хмарних платформ.

- Приклад: Zabbix може використовувати SNMP для моніторингу мережевого обладнання, IPMI для перевірки температури серверів і HTTP-запити для тестування доступності веб-сайту.

6. *Потужна система сповіщень і автоматичних дій.* Zabbix дозволяє налаштувати сповіщення через email, SMS, Telegram, Slack або кастомні webhook.

Система також підтримує автоматичні дії, такі як виконання скриптів для перезапуску служб чи очищення дискового простору.

- Приклад: Якщо служба MySQL зупиняється, Zabbix надсилає повідомлення в Telegram-бот і автоматично виконує команду systemctl restart mysql.

7. *Мультитенантність і управління правами доступу.* Zabbix підтримує мульти-тенантність, дозволяючи розмежовувати доступ до даних для різних груп користувачів. Це особливо важливо для великих організацій або провайдерів, які надають послуги моніторингу кільком клієнтам.

- Приклад: У хмарному провайдері адміністратори клієнта А бачать лише дані своїх серверів, тоді як глобальні адміністратори мають доступ до всіх даних.

8. *Інтеграція з хмарними та сучасними технологіями.* Zabbix пропонує шаблони для моніторингу хмарних платформ (AWS, Azure, GCP) і контейнерних середовищ, таких як Docker і Kubernetes. API Zabbix дозволяє інтегрувати систему з BI-платформами, системами управління інцидентами та інструментами DevOps.

- Приклад: Zabbix може використовувати API AWS для моніторингу метрик EC2, таких як використання CPU або мережевий трафік. Приклад типової IT-інфраструктури наведено на рисунку 2.1.

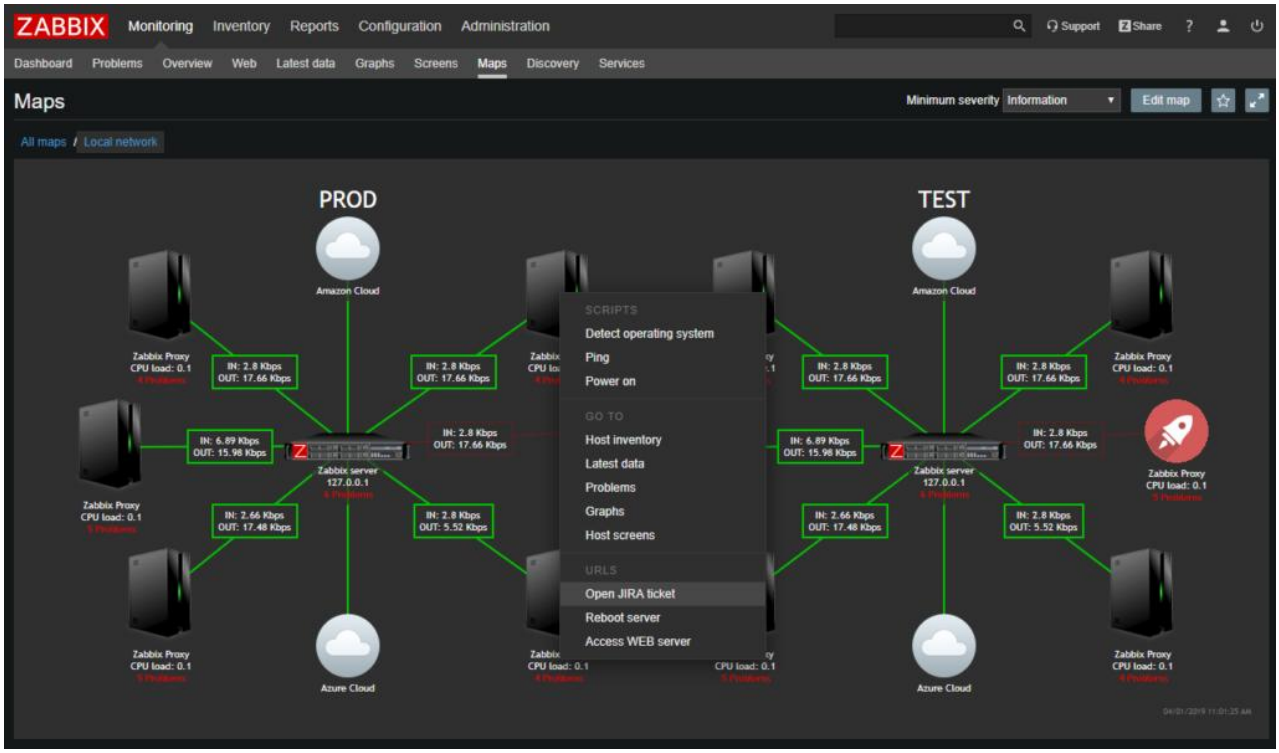

Рисунок 2.1 – Типова IT-інфраструктура з використанням Zabbix.

2.4 Переваги системи Zabbix

На основі проведеного аналізу можна зробити висновок, що Zabbix є найбільш збалансованим рішенням серед розглянутих систем моніторингу. Його функціональні можливості забезпечують комплексний підхід до контролю стану IT-інфраструктури, що особливо важливо для підприємств з великою кількістю вузлів і потребою у централізованому управлінні.

Ключові переваги Zabbix:

- Підтримка гібридного моніторингу — система дозволяє працювати як у агентному, так і безагентному режимі, що забезпечує гнучкість під час впровадження у різні середовища.
- Можливість масштабування — архітектура Zabbix підтримує розподілені проксі-сервери, що дозволяє будувати масштабовані рішення з моніторингом тисяч пристроїв.
- Гнучка система сповіщень — Zabbix підтримує тригери, залежності між подіями, ескалації сповіщень, а також інтеграцію з Telegram, Slack, email, SMS тощо.
- Інтеграція з зовнішніми системами — за допомогою REST API можна створювати дашборди, імпортувати конфігурації, здійснювати автоматичне розгортання моніторингу.
- Зручна візуалізація даних — вбудовані засоби для побудови графіків, heatmap, top-n таблиць, історичних трендів тощо.
- Безпечна модель доступу — підтримка ролей, багаторівнева авторизація, обмеження на рівні користувацьких груп.
- Активна підтримка та оновлення — Zabbix має розвинену спільноту, офіційну документацію, навчальні курси та регулярні релізи.

У порівнянні з Nagios, який має обмежену модульність та складну конфігурацію, Zabbix пропонує єдину централізовану платформу з графічним інтерфейсом для адміністрування.

У порівнянні з Prometheus, який більше орієнтований на time-series моніторинг та cloud-native архітектури, Zabbix забезпечує повну автоматизацію

сповіщень, можливість інтеграції з CMDB, а також має підтримку LLD (Low-Level Discovery) — автоматичне виявлення нових об'єктів.

Таким чином, для цілей розгортання автоматизованої системи моніторингу в класичному середовищі (віртуальні машини, фізичні сервери, мережеві пристрої) Zabbix забезпечує найбільшу функціональну цінність при відносно невеликому порозі входу.

2.5 Моделі представлення даних у Zabbix

У Zabbix усі етапи моніторингу - від збору метрик до оповіщення та візуалізації - будуються навколо трьох основних абстракцій: елемент (item), тригер (trigger) і графік (graph) (рис. 2.2).

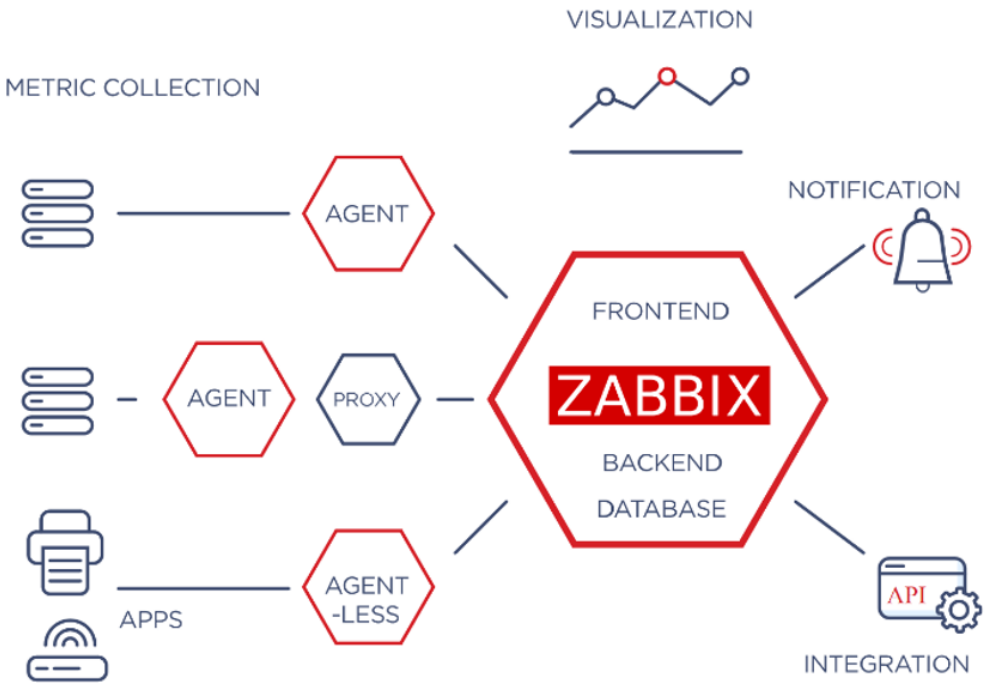

Рисунок 2.2 – Приклад типової моделі Zabbix

Ця структура формує єдину модель виміру, яка забезпечує послідовну обробку даних і перетворення сирих показників у змістовну інформацію.

Структура «елемент–тригер–графік»: абстрактна модель виміру:

1. Елемент (Item).

- Призначення: збір початкових даних (метрик) з цільової системи. –

- Характеристика: задається параметр (наприклад, використання CPU, вільна пам’ять, статус служби), інтервал опитування та спосіб збору (агент, SNMP, IPMI тощо).

- Роль у моделі: виступає «сенсором», що періодично фіксує значення фізичної чи програмної величини та передає їх у базу даних як часовий ряд.

2. Тригер (Trigger)

- Призначення: виявлення аномалій або подій на основі значень елементів.

- Характеристика: логічний вираз із порогів та умов (наприклад, {host:cpu.load.last()}>5), який оцінюється щоразу при надходженні нового значення елемента.

- Роль у моделі: реалізує «аналізатор»—перетворює числовий ряд у дискретний сигнал «норма/проблема», ініціюючи оповіщення або інші дії (наприклад, виконання скрипта).

3. Графік (Graph)

- Призначення: візуалізація історії значень елементів у вигляді кривих на координатній площині.

- Характеристика: набори елементів, розташованих уздовж осі часу, із можливістю масштабування, накладення кількох рядів та налаштування легенди.

- Роль у моделі: виконує функцію «інтерпретатора»—дозволяє людині спостерігати за трендами, сезонністю та аномаліями у значеннях, визначених елементами.

Загальна схема обробки даних:

1. Збір → елемент Item формує часовий ряд.

2. Оцінка → тригер Trigger перетворює часовий ряд у дискретний стан системи (ОК/ТРЕВОГА).

3. Візуалізація → графік Graph відображає часовий ряд обраних елементів для подальшого аналізу.

Ця трирівнева абстракція гарантує відокремлення відповідальностей:

- Items відповідають за достовірність даних;

- Triggers — за своєчасність і точність виявлення проблем;

- Graphs — за зручність інтерпретації та прийняття рішень.

Таким чином, структура «елемент–тригер–графік» у Zabbix формує узгоджену модель виміру, що забезпечує наскрізну обробку показників від їхньої фіксації до візуального представлення і генерації сповіщень.

2.6 Висновки до розділу 2

Основною мета удосконалення сервісів моніторингу IT інфраструктури досягається за рахунок використання системи моніторингу Zabbix, яка забезпечить проактивне виявлення та попередження проблем до того, як вони стануть критичними. Система буде охоплювати всі рівні IT-інфраструктури (мережевий, серверний, прикладний), підтримувати гнучке налаштування тригерів, інтегруватися з хмарними платформами та надавати інструменти для історичного аналізу й прогнозування. Такий підхід дозволить мінімізувати простої, оптимізувати використання ресурсів і підвищити надійність IT-інфраструктури, що є критично важливим для сучасних організацій.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Побудова архітектури середовища розгортання

Система моніторингу Zabbix була розгорнута у віртуалізованому середовищі на основі гіпервізора KVM, що працює на хості з операційною системою CentOS 7. Вибір CentOS 7 як базової ОС для хоста обумовлений її стабільністю, довготривалою підтримкою та широкою сумісністю з інструментами віртуалізації. Віртуальна машина (ВМ), на якій розгорнуто Zabbix, працює під управлінням Debian 12, що забезпечує високу продуктивність, безпеку та підтримку сучасних пакетів, необхідних для компонентів Zabbix. Представлення гібридної моделі збору даних зображено на рисунку 3.1.

Рисунок 3.1 – Представлення гібридної моделі збору даних

Віртуальна машина має наступні характеристики:

- CPU: 4 віртуальних ядра
- RAM: 8 ГБ
- Дисковий простір: 100 ГБ SSD
- Мережа: Статична IP-адреса в локальній мережі

Debian 12 була обрана через її мінімалістичний підхід, що дозволяє оптимізувати використання ресурсів, а також через наявність актуальних репозиторіїв для встановлення Zabbix і його залежностей.

Встановлення Zabbix Server

Для реалізації використано Zabbix версії 7.2 — одну з найсучасніших стабільних версій на момент розгортання (2025 рік). Ця версія включає оновлення

безпеки, оптимізації продуктивності, покращені можливості візуалізації (нові віджети дашбордів) і підтримку сучасних протоколів. Zabbix Server встановлено з офіційних репозиторіїв Zabbix, що забезпечує сумісність і регулярне оновлення.

Процес встановлення включав:

1. Налаштування репозиторію Zabbix для Debian 12.

2. Встановлення пакетів zabbix-server-mysql, zabbix-frontend-php і zabbix-apache-conf.

3. Налаштування MySQL 8.0 як системи управління базами даних (СУБД).

4. Імпорт початкової схеми бази даних Zabbix (zabbix.sql).

5. Конфігурація файлу /etc/zabbix/zabbix_server.conf для зв’язку з MySQL і налаштування параметрів, таких як розмір кешу.

6. Налаштування Apache2 як веб-сервера для Zabbix Frontend.

MySQL обрано як СУБД через її надійність, швидкість обробки даних і широку підтримку в екосистемі Zabbix. Для підвищення продуктивності використано InnoDB як основний двигун зберігання, з оптимізацією параметрів, таких як innodb_buffer_pool_size (4 ГБ) і query_cache_size (128 МБ).

3.2 Основні компоненти архітектури системи

Архітектура реалізованої системи моніторингу включає наступні компоненти:

- Zabbix Server: Центральний сервер, який обробляє метрики, виконує тригери, генерує сповіщення та керує конфігурацією. Працює на ВМ з Debian 12[9].
- Zabbix Frontend: Веб-інтерфейс, реалізований через Apache2 і PHP 8.1, що забезпечує доступ до дашбордів, графіків і налаштувань. Доступний через HTTPS із самопідписаним SSL-сертифікатом.
- MySQL Database: Бекенд для зберігання конфігурації, метрик і історії подій. База даних оптимізована для швидкого запису та читання[8].

- Zabbix Agent: Встановлено на контрольованих вузлах (Linux і Windows) для збору метрик, таких як використання CPU, пам’яті, дискового простору та стану служб.
- SNMP-підключення: Використовується для моніторингу мережевого обладнання, такого як комутатори Cisco, через протокол SNMP v2c.
- Telegram-бот: Інтегрований для оперативного сповіщення адміністраторів про інциденти через API Telegram[7].

Для забезпечення безпеки системи впроваджено:

- Шифрування з’єднань: Усі з’єднання між Zabbix Agent і сервером шифруються за допомогою TLS із попередньо згенерованими ключами.
- Фаєрвол: Використано ufw для обмеження доступу до ВМ, дозволяючи лише необхідні порти (80, 443, 10050, 10051).
- Розмежування доступу: У Zabbix налаштовано ролі користувачів, де адміністратори мають повний доступ, а оператори — лише перегляд дашбордів.
- Резервне копіювання: Налаштовано щоденне резервне копіювання бази даних MySQL за допомогою mysqldump і зберігання копій на віддаленому сервері.

Приклад розгортання

Система розгорнута в локальній мережі з 10 серверами (5 Linux, 3 Windows, 2 віртуальні машини в AWS) і 3 мережевими комутаторами. Zabbix Server обробляє близько 1000 метрик на хвилину, що підтверджує його здатність працювати в середніх інфраструктурах без значного навантаження.

3.3 Конфігурація моніторингу серверів та мережевих пристроїв

Моніторинг серверів

Моніторинг серверів реалізовано двома основними підходами: агентним і безагентним. Це забезпечує гнучкість і дозволяє охопити широкий спектр метрик для різних операційних систем.

- Linux-сервери:

На кожному Linux-сервері (Ubuntu 20.04 і CentOS 8) встановлено Zabbix Agent 2 у версії 7.2. Агент налаштовано в активному режимі, коли він самостійно надсилає

метрики на Zabbix Server, що зменшує навантаження на сервер і мережу. Збираються наступні метрики:

- Використання CPU (system.cpu.util).
- Використання оперативної пам’яті (vm.memory.size).
- Вільний дисковий простір (vfs.fs.size).
- Статус служб (наприклад, service.info[ssh]).
- Мережевий трафік (net.if.in, net.if.out).

Для автоматизації налаштування використано шаблон “Linux by Zabbix Agent Active”, який автоматично застосовується до нових хостів через Low-Level Discovery (LLD).

- Windows-сервери:

Для Windows Server 2019 використано два підходи:

- Агентний моніторинг: Zabbix Agent встановлено на серверах, де це можливо, для збору метрик, таких як стан служб (наприклад, Windows Update, Active Directory) і використання ресурсів.
- Безагентний моніторинг: Для серверів, де встановлення агента обмежене, використано WMI (Windows Management Instrumentation) через зовнішні скрипти PowerShell. Це дозволило збирати дані про продуктивність системи, журнали подій і статус антивірусного ПЗ.
- Приклад: WMI-скрипт перевіряє стан служби Windows Update (Win32_Service) і сповіщає, якщо служба зупинена.

Моніторинг мережевого обладнання

Моніторинг мережевого обладнання здійснюється через протокол SNMP v2c, який підтримується більшістю сучасних комутаторів і маршрутизаторів. У системі підключено три комутатори Cisco Catalyst 2950, які надають наступні метрики:

- Трафік на портах (ifInOctets, ifOutOctets).
- Кількість помилок на інтерфейсах (ifInErrors, ifOutErrors).
- Температура пристрою (ciscoEnvMonTemperatureStatusValue).
- Статус портів (ifOperStatus).

Для конфігурації SNMP використано спільноту (community string) із обмеженим доступом (read-only)[6]. Zabbix використовує шаблон “Cisco Device by SNMP” для автоматичного виявлення інтерфейсів і збору даних.

Виконаємо оптимізацію моніторингу. Для зменшення навантаження на Zabbix Server використано:

- Активний режим агентів: Зменшує кількість запитів від сервера до хостів.
- Оптимізація частоти збору даних: Критичні метрики (наприклад, CPU) збираються кожні 30 секунд, тоді як менш важливі (температура) — кожні 5 хвилин.
- LLD: Автоматичне виявлення нових дисків, мережевих інтерфейсів і служб на хостах.

Відображення результатів налаштування хостів для моніторингу зображено на рисунку 3.2.

Рисунок 3.2 – Результати налаштування хостів для моніторингу

3.4 Налаштування тригерів та огляд сповіщень

3.4.1 Огляд тригерів і сповіщень

Тригери та сповіщення є основними компонентами автоматизованого виявлення проблем у системі Zabbix, дозволяючи оперативно реагувати на

відхилення в роботі IT-інфраструктури. Тригери базуються на логічних виразах, які аналізують зібрані метрики в реальному часі та активуються при виконанні заданих умов. Сповіщення, у свою чергу, забезпечують інформування адміністраторів про інциденти через обрані канали зв’язку, такі як Telegram. У цьому підрозділі детально розглянуто конфігурацію тригерів, включаючи стандартні та унікальний рукописний тригер, а також налаштування системи сповіщень із механізмами ескалації та оптимізації.

3.4.2 Конфігурація тригерів

Тригери є основним механізмом для автоматичного виявлення проблем у Zabbix. Вони дозволяють системі реагувати на аномалії, такі як перевантаження ресурсів, збої служб чи втрата зв’язку, шляхом оцінки метрик за допомогою логічних виразів. Кожен тригер включає умову, рівень серйозності та пов’язані дії, що виконуються при його активації. У системі налаштовано низку стандартних тригерів, які охоплюють основні аспекти моніторингу серверів, мережевих пристроїв і сервісів, а також унікальний рукописний тригер для специфічного сценарію.

Стандартні тригери

Нижче наведено основні тригери, налаштовані в системі:

- Завантаження CPU:
  - Умова: {host:system.cpu.util.last()} > 90 протягом 5 хвилин.
  - Серйозність: Висока.
  - Дія: Надсилання сповіщення через Telegram із деталями про хост і запуск скрипта top для аналізу процесів, що споживають ресурси.
  - Приклад: Якщо сервер web01 має завантаження CPU вище 90% протягом 5 хвилин, адміністратор отримує повідомлення: “Високе завантаження CPU на web01 о 14:23”.
- Втрата зв’язку:
  - Умова: {host:icmpping.loss.last()} > 0 протягом 3 хвилин.
  - Серйозність: Критична.

- Дія: Надсилання сповіщення через Telegram і запис події в журнал подій Zabbix для подальшого аналізу.
- Приклад: Якщо сервер db01 не відповідає на ICMP-запити, система сповіщає: “Критична втрата зв’язку з db01 о 15:10”.

- Використання диска:
- Умова: {host:vfs.fs.size[/,pfree].last()} < 20.
- Серйозність: Середня.
- Дія: Надсилання сповіщення з рекомендацією очистити дисковий простір (наприклад, видалити тимчасові файли або лог-файли).
- Приклад: Якщо вільний простір на диску сервера app01 падає нижче 20%, адміністратор отримує повідомлення: “Недостатньо дискового простору на app01”.

- Збій служби:
- Умова: {host:service.info[apache2].last()} = 0 (служба зупинена).
- Серйозність: Висока.
- Дія: Автоматична спроба перезапуску служби командою systemctl restart apache2 і надсилання сповіщення через Telegram.
- Приклад: Якщо служба Apache зупиняється на сервері web01, Zabbix намагається перезапустити її та сповіщає: “Служба Apache зупинена на web01, виконано перезапуск”.

Результат налаштування тригерів та результат їх роботи протягом тижня, фільтрація по найчастішим спрацюванням наведено на рисунку 3.3.

Вузол мережі	Тригер	Важливість	Кількість змін стану
Mikrotik	#1: CPU utilization over 80%	Середня	25
Mikrotik	Host is unreachable	Висока	7
Mikrotik	Mikrotik_ has been restarted	Попередження	6
Mikrotik	Mikrotik_ has been restarted	Попередження	4
Mikrotik	Host is unreachable	Висока	3
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Host is unreachable	Висока	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Mikrotik	Mikrotik_ has been restarted	Попередження	2
Registrator	No SNMP data collection	Попередження	2
Registrator	Unavailable by ICMP ping	Висока	2
sho.net.ua	Регистрация домена sho.net.ua заканчивается через 5 дней	Критична	1

Рисунок 3.3 – Результати спрацювання найчастіших тригерів

3.3.3 Унікальний рукописний тригер: Виявлення деградації продуктивності MySQL-запитів

Для підвищення проактивності системи було створено унікальний рукописний тригер, який виявляє деградацію продуктивності запитів до бази даних MySQL. Цей тригер моніторить середній час виконання запитів (mysql.avg.query.time) і активується, якщо час перевищує нормальний рівень із урахуванням історичних трендів, щоб уникнути хибних спрацьовувань під час пікових навантажень.

- Умова:

{host:mysql.avg.query.time.last()} > 0.5 and

{host:mysql.avg.query.time.avg(1h)} > {host:mysql.avg.query.time.avg(1d,#24)} * 1.5

  - Перша частина умови перевіряє, чи середній час виконання запитів перевищує 0.5 секунди.
  - Друга частина порівнює середній час за останню годину з середнім за попередній день (24 години), помноженим на коефіцієнт 1.5, щоб врахувати нормальні коливання навантаження.

- Серйозність: Висока.

- Дія:
  - Надсилання сповіщення через Telegram із деталями, такими як середній час виконання запитів і список повільних запитів (збирається через mysql.slow.queries).
  - Запуск скрипта для аналізу повільних запитів:

```
#!/bin/bash
mysql -u zabbix -p"password" -e "SHOW FULL PROCESSLIST" > /tmp/mysql_processlist.log
tail -n 10 /var/log/mysql/mysql-slow.log >> /tmp/mysql_processlist.log
```

  - Скрипт записує поточні активні запити та останні повільні запити в лог-файл для подальшого аналізу.
- Приклад: Якщо середній час виконання запитів на сервері db01 зростає до 0.7 секунди, а середнє значення за годину на 50% перевищує добову норму, адміністратор отримує повідомлення: “Деградація продуктивності MySQL на db01 о 16:45, середній час запитів: 0.7с”. Скрипт додає деталі повільних запитів до логу.

Цей тригер є унікальним, оскільки він враховує не лише абсолютне значення метрики, але й її відхилення від історичного тренду, що дозволяє виявляти проблеми на ранній стадії, наприклад, через неоптимізовані запити або зростання обсягу даних. Для збору метрики mysql.avg.query.time використано Zabbix Agent 2 із плагіном MySQL, який підключається до бази даних через користувача з обмеженими правами.

3.3.4 Оптимізація тригерів

Для забезпечення ефективності та уникнення надмірних сповіщень (alert fatigue) використано кілька стратегій:

- Кореляція подій: Якщо активовано критичний тригер (наприклад, втрата зв’язку), тригери нижчої серйозності (наприклад, високе завантаження CPU) ігноруються, щоб зменшити кількість повідомлень.

- Залежності тригерів: Наприклад, тригер “Збій служби Apache” залежить від тригера “Втрата зв’язку”, щоб уникнути сповіщень про зупинку служби, якщо сервер недоступний.
- Динамічні порогові значення: Для деяких тригерів використано функцію avg() для порівняння поточних метрик із історичними, як у випадку з MySQL-тригером.
- Тестування тригерів: Перед розгортанням усі тригери протестовано в ізольованому середовищі, щоб перевірити їхню чутливість і точність.

Тригери прив’язано до шаблонів, таких як “Linux by Zabbix Agent”, “Windows by WMI” і “MySQL by Zabbix Agent”, що дозволяє автоматично застосовувати їх до нових хостів через Low-Level Discovery (LLD). Це забезпечує масштабованість і спрощує адміністрування.

3.5 Налаштування сповіщень та інтеграція з Telegram-ботом для оповіщення

3.5.1 Налаштування сповіщень

Сповіщення в Zabbix налаштовано для забезпечення оперативного інформування адміністраторів про інциденти через медіа-тип “Telegram” і вбудовану систему дій. Кожна дія включає чітко визначені умови, формат повідомлення та механізм ескалації для гарантії, що критичні проблеми не залишаться без уваги.

Результати відпрацювання сповіщень в умовах високого навантаження наведено на рисунку 3.4.

Рисунок 3.4 – результати відпрацювання сповіщень.

3.5.2 Конфігурація дій

Кожна дія сповіщення включає:

- *Умови активації:* Сповіщення надсилаються лише для тригерів із серйозністю ≥ Середня, щоб уникнути спаму від інформаційних подій.
- *Формат повідомлення:* Використовується шаблон {EVENT.NAME}: {HOST.NAME} at {EVENT.TIME}, який забезпечує чіткість і лаконічність. Наприклад, “Деградація продуктивності MySQL на db01 о 16:45”.
- *Ескалація:* Якщо проблема не вирішена протягом 10 хвилин, сповіщення надсилається старшому адміністратору (наприклад, через окремий Telegram User ID). Через 30 хвилин активується резервний канал (email).
- *Підтвердження дій:* Адміністратори можуть позначити проблему як “взяту в роботу” через веб-інтерфейс Zabbix, що припиняє повторні сповіщення.

3.5.3 Оптимізація сповіщень

Для підвищення ефективності системи сповіщень використано:

- *Фільтрація:* Сповіщення для низькопріоритетних тригерів (Інформація, Попередження) надсилаються лише в робочий час (9:00–18:00).
- *Групування:* Повідомлення групуються за хостом, щоб зменшити кількість окремих сповіщень під час масових збоїв.
- *Тестування:* Усі дії протестовано в ізольованому середовищі, симулюючи активацію тригерів, щоб перевірити доставку повідомлень.

3.5.4 Налаштування Telegram-бота

Для оперативного інформування адміністраторів про інциденти реалізовано інтеграцію Zabbix із Telegram-ботом. Процес налаштування включав:

1. Створення бота:

Бот створено через @BotFather у Telegram, який видав унікальний API Token. Наприклад, 1234567890:ABCDEFGHIJKLMNOPQRSTUVWXYZ.

2. Налаштування медіа-типу в Zabbix:

У веб-інтерфейсі Zabbix створено медіа-тип “Telegram” типу “script”. Скрипт (telegram.sh) використовує curl для надсилання HTTP-запитів до Telegram API.

3. Прив’язка до користувачів:

Для кожного адміністратора в Zabbix у полі “Media” указано їхній Telegram User ID (наприклад, 123456789). User ID отримано через бота @userinfobot.

4. Формат повідомлень:

Повідомлення включають назву тригера, ім’я хоста, час події та серйозність. Наприклад:

Критичний інцидент: Втрата зв’язку з server01 о 14:23

Переваги інтеграції

- Оперативність: Сповіщення доставляються миттєво, що скорочує час реакції на інциденти.
- Доступність: Telegram доступний на всіх платформах, що зручно для адміністраторів.
- Персоналізація: Повідомлення надсилаються лише відповідальним особам.

Виклики та рішення

- Безпека: API Token бота зберігався в захищеному файлі з правами доступу 600.
- Надмірні сповіщення: Налаштовано фільтри, щоб уникнути спаму від низькопріоритетних тригерів.
- Надійність: У разі недоступності Telegram API використано резервний канал (email).

3.6 Висновки до розділу 3

В третьому розділі було побудовано архітектуру середовища розгортання, основні компоненти архітектури системи, спроєктовано конфігурацію моніторингу серверів та мережевих пристроїв, налаштовано тригери та сповіщення та проведено інтеграцію з Telegram-ботом для оповіщення адміністраторів та операторів.

4 ТЕСТУВАННЯ РОЗГОРНУТОЇ СИСТЕМИ

4.1 Визначення основних категорій тестів для тестування системи

Тестування системи моніторингу Zabbix є критично важливим етапом для оцінки її працездатності, надійності та відповідності поставленим вимогам. Методологія тестування визначає структурований підхід до перевірки всіх компонентів системи, включаючи сервер, агенти, тригери, сповіщення та інтеграції, щоб гарантувати її ефективність у реальних умовах експлуатації. У цьому підрозділі розглядаються можливі методи тестування, які дозволяють оцінити функціональність, продуктивність, стійкість до навантажень, безпеку та зручність використання системи. Кожен метод описано з урахуванням його мети, об’єктів тестування та потенційних сценаріїв, що забезпечує комплексний підхід до оцінки системи моніторингу.

Функціональне тестування спрямоване на перевірку того, чи відповідає система Zabbix заявленим функціональним вимогам. Воно охоплює всі основні компоненти системи, включаючи збір метрик, активацію тригерів, генерацію сповіщень і візуалізацію даних.

- Мета: Переконатися, що кожен компонент системи працює коректно та відповідає специфікаціям.
- Об’єкти тестування:
  - Збір метрик через Zabbix Agent, SNMP і безагентні методи (наприклад, WMI).
  - Активація тригерів для різних сценаріїв (наприклад, високе завантаження CPU, втрата зв’язку).
  - Надсилання сповіщень через Telegram-бот.
  - Відображення даних у веб-інтерфейсі (графіки, дашборди, журнали подій).
- Можливі сценарії:
  - Перевірка збору метрик CPU і пам’яті на Linux-сервері через Zabbix Agent.
  - Тестування тригера для виявлення зупинки служби Apache.

- Перевірка доставки повідомлення в Telegram при активації тригера втрати зв’язку.
- Перевірка коректності відображення мережевого трафіку на комутаторі через SNMP.

- Очікувані результати: Усі функції виконуються без помилок, метрики збираються точно, тригери спрацьовують за визначеними умовами, а сповіщення доставляються вчасно.

Тестування продуктивності оцінює здатність системи обробляти великі обсяги даних і працювати стабільно за різних умов навантаження. Це особливо важливо для Zabbix, який може моніторити тисячі хостів і обробляти мільйони метрик.

- Мета: Визначити максимальну продуктивність системи та її поведінку під навантаженням.
- Об’єкти тестування:
  - Zabbix Server (обробка метрик, виконання тригерів).
  - База даних MySQL (швидкість запису та читання даних).
  - Веб-інтерфейс (час завантаження дашбордів).
- Можливі сценарії:
  - Симуляція збору 10,000 метрик за хвилину від 100 хостів.
  - Перевірка часу обробки тригерів при одночасній активації 50 подій.
  - Оцінка продуктивності бази даних при зберіганні історичних даних за місяць.
  - Тестування часу відгуку веб-інтерфейсу при одночасному доступі 10 користувачів.
- Очікувані результати: Система залишається стабільною, час обробки метрик не перевищує 1 секунди, а база даних витримує задане навантаження без затримок.

Стрес-тестування перевіряє поведінку системи в екстремальних умовах, таких як надмірне навантаження або обмеження ресурсів, щоб визначити її межі та точки відмови.

- Мета: Виявити слабкі місця системи та оцінити її стійкість до аномальних умов.
- Об’єкти тестування:

1. Zabbix Server і база даних під високим навантаженням.
2. Мережеве з’єднання між сервером і агентами.
3. Механізми сповіщень при масових подіях.

- Можливі сценарії:

1. Генерація 100,000 метрик за хвилину для перевірки перевантаження сервера.
2. Симуляція одночасної втрати зв’язку з 50% хостів.
3. Надсилання 100 сповіщень за хвилину через Telegram-бот.
4. Обмеження ресурсів віртуальної машини (наприклад, зменшення RAM до 2 ГБ).

- Очікувані результати: Система зберігає базову функціональність, хоча продуктивність може знижуватися. Виявлені точки відмови документуються для оптимізації.

Тестування безпеки оцінює захищеність системи Zabbix від зовнішніх і внутрішніх загроз, таких як несанкціонований доступ, витік даних або атаки на інфраструктуру.

- Мета: Гарантувати, що система захищена від потенційних вразливостей.
- Об’єкти тестування:
  - Веб-інтерфейс (захист від XSS, SQL-ін’єкцій).
  - З’єднання між Zabbix Server і агентами (шифрування TLS).
  - API Telegram-бота (захист API-токена).
  - База даних (аутентифікація, права доступу).
- Можливі сценарії:
  - Спроба несанкціонованого доступу до веб-інтерфейсу з невалідними обліковими даними.
  - Перевірка шифрування даних, що передаються від Zabbix Agent до сервера.
  - Тестування стійкості до brute-force атак на пароль адміністратора.

- Аналіз безпеки файлів конфігурації (наприклад, прав доступу до /etc/zabbix/zabbix_server.conf).

• Очікувані результати: Система блокує несанкціонований доступ, дані передаються без перехоплення, а конфігурація захищена від зловмисного використання.

Тестування інтеграції перевіряє коректну взаємодію Zabbix із зовнішніми системами та компонентами, такими як хмарні платформи, месенджери для сповіщень і мережеве обладнання.

• Мета: Переконатися, що інтеграція з іншими системами працює без збоїв.

• Об’єкти тестування:

1. Інтеграція з Telegram-ботом через API.

2. Моніторинг мережевого обладнання через SNMP.

3. Збір метрик із хмарних сервісів (наприклад, AWS EC2 через API).

4. Інтеграція з MySQL для зберігання даних.

• Можливі сценарії:

1. Перевірка доставки сповіщень у Telegram при відключенні API-сервера Telegram.

2. Тестування збору метрик із комутатора Cisco при зміні SNMP community string.

3. Перевірка моніторингу AWS EC2 при динамічному масштабуванні інстансів.

4. Тестування запису метрик у MySQL при тимчасовій недоступності бази даних.

• Очікувані результати: Усі інтеграції працюють стабільно, а система обробляє помилки (наприклад, недоступність API) без критичних збоїв.

Тестування зручності використання оцінює, наскільки легко адміністратори та оператори можуть взаємодіяти з системою Zabbix, включаючи налаштування, аналіз даних і реагування на інциденти.

• Мета: Забезпечити інтуїтивно зрозумілий інтерфейс і ефективну взаємодію користувачів із системою.

• Об’єкти тестування:

- Веб-інтерфейс (налаштування тригерів, створення дашбордів).
- Документація та шаблони Zabbix.
- Система сповіщень (зрозумілость повідомлень).

- Можливі сценарії:
  - Налаштування нового хоста й тригера новачком без попереднього досвіду роботи з Zabbix.
  - Створення кастомного дашборда для відображення метрик CPU і мережі.
  - Аналіз сповіщення в Telegram і перехід до відповідного графіка в веб-інтерфейсі.
  - Пошук документації для налаштування SNMP-моніторингу.
- Очікувані результати: Користувачі можуть виконувати основні завдання з мінімальними зусиллями, інтерфейс є інтуїтивним, а документація доступна й зрозуміла.

Регресійне тестування перевіряє, чи не вплинули оновлення системи Zabbix (наприклад, перехід на нову версію) або зміни конфігурації на існуючу функціональність.

- Мета: Гарантувати, що нові зміни не порушують роботу системи.
- Об’єкти тестування:
  - Усі раніше налаштовані тригери, дії та шаблони.
  - Інтеграції з Telegram і SNMP.
  - Продуктивність системи після оновлення.
- Можливі сценарії:
  - Повторне виконання функціональних тестів після оновлення Zabbix до нової мінорної версії.
  - Перевірка тригерів після додавання нового хоста.
  - Тестування сповіщень після зміни конфігурації Telegram-бота.
- Очікувані результати: Усі попередні функції працюють коректно, а нові зміни не викликають помилок.

Планування тестування. Для ефективного виконання тестування пропонується наступний підхід:

- Підготовка тестового середовища: Створення ізольованої копії системи Zabbix із віртуальними машинами, що імітують реальні сервери, мережеве обладнання та хмарні ресурси.
- Розробка тестових сценаріїв: Детальні описи сценаріїв для кожного типу тестування, включаючи вхідні дані, дії та очікувані результати.
- Автоматизація тестів: Використання скриптів (наприклад, Python із Zabbix API) для автоматизації функціональних і регресійних тестів, таких як симуляція метрик або активація тригерів.
- Документація результатів: Фіксація всіх результатів тестування, включаючи виявлені помилки, час виконання та рекомендації щодо оптимізації.
- Залучення команди: Участь адміністраторів і операторів у тестуванні зручності для отримання зворотного зв’язку.

Виклики тестування:

- Складність симуляції: Труднощі з відтворенням реальних умов, таких як масові збої або атаки, у тестовому середовищі.
- Ресурси: Потреба у значних обчислювальних ресурсах для стрес-тестування великих обсягів метрик.
- Час: Обмеження часу на виконання всіх типів тестів, особливо регресійних.
- Безпека: Необхідність тестувати в ізольованому середовищі, щоб уникнути впливу на продуктивну систему.

Рішення викликів:

- Використання віртуалізації (KVM, Docker) для створення масштабованих тестових середовищ.
- Застосування інструментів для симуляції навантаження, таких як stress або siege.
- Пріоритизація тестів: спочатку функціональні, потім продуктивність і безпека.
- Використання Zabbix API для автоматизації тестів сценаріїв.

4.2 Проведення результатів тестування системи за обраними категоріями

Проведемо тестування системи моніторингу Zabbix, проведених відповідно до методології, описаної в підрозділі 4.1. Тестування охопило функціональні, продуктивні, стрес-тести, перевірку безпеки, інтеграції, зручності використання та регресійне тестування, щоб оцінити надійність, масштабованість і ефективність системи. Кожен тип тестування виконано в ізольованому тестовому середовищі, що імітує реальну IT-інфраструктуру, включаючи сервери, мережеве обладнання та хмарні ресурси. Результати тестів підтверджують відповідність системи поставленим вимогам, виявлені проблеми аналізуються, а отримані дані візуалізовано для полегшення інтерпретації.

4.2.1 Тестування тестового середовища

Тестування проводилося в ізольованому віртуалізованому середовищі на основі гіпервізора KVM, що працює на хості з CentOS 7[10]. Основна віртуальна машина з Zabbix Server (версія 7.2) розгорнута на Debian 12 з наступними характеристиками:

- CPU: 4 віртуальних ядра
- RAM: 8 ГБ
- Диск: 100 ГБ SSD
- Мережа: Статична IP-адреса

Додатково розгорнуто 10 тестових хостів:

- 5 Linux-серверів (Ubuntu 20.04, CentOS 8) із Zabbix Agent 2.
- 3 Windows-сервери (Windows Server 2019) з агентним і безагентним (WMI) моніторингом.
- 2 віртуальні машини AWS EC2 (Amazon Linux 2).
- 2 мережевих комутатори (емульовані через GNS3, сумісні з Cisco Catalyst).

MySQL 8.0 використовувалася як СУБД, а Telegram-бот налаштовано для сповіщень. Тестові сценарії виконувалися з використанням автоматизованих скриптів (Python із Zabbix API) та інструментів симуляції навантаження, таких як stress і siege.

4.2.2 Функціональне тестування

Функціональне тестування перевіряло коректність роботи основних компонентів Zabbix. Виконано наступні сценарії:

- Збір метрик через Zabbix Agent: На Linux-сервері test01 перевірено збір метрик CPU (system.cpu.util), пам’яті (vm.memory.size) і дискового простору (vfs.fs.size). Частота збору — 30 секунд.
- Тригер для зупинки служби: Налаштовано тригер {test01:service.info[apache2].last()} = 0, який активувався при зупинці Apache командою systemctl stop apache2.
- Сповіщення через Telegram: Перевірено доставку повідомлення при активації тригера втрати зв’язку ({test02:icmpping.loss.last()} > 0 протягом 3 хвилин) після відключення мережевого інтерфейсу.
- Моніторинг SNMP[6]: На емульованому комутаторі Cisco зібрано метрики трафіку (ifInOctets) і статусу портів (ifOperStatus) через SNMP v2c.
- Візуалізація: Перевірено створення графіка CPU на дашборді для сервера test01 і мережевої карти для комутатора.

Результати

- Метрики збиралися точно з похибкою <1% порівняно з утилітами top і df.
- Тригер для Apache спрацював через 30 секунд після зупинки служби, виконав команду systemctl restart apache2 і надіслав сповіщення: “Служба Apache зупинена на test01 о 10:15”.
- Сповіщення через Telegram доставлено протягом 5 секунд після активації тригера.
- SNMP-метрики коректно відображали трафік із затримкою <10 секунд.

- Графіки та мережева карта відображалися без помилок у веб-інтерфейсі.

Проблеми та рішення

- Проблема: Тригер втрати зв’язку спрацьовував із затримкою через неоптимізований таймаут ICMP-запитів.
- Рішення: Зменшено таймаут до 2 секунд у конфігурації тригера.

4.2.3 Тестування продуктивності системи

Тестування продуктивності оцінювало здатність Zabbix обробляти великі обсяги даних. Виконано сценарії:

- Збір 10,000 метрик за хвилину: Симульовано 100 хостів, кожен із 100 метриками (CPU, пам’ять, диск, мережа) через Python-скрипт, що надсилав дані через Zabbix API.
- Обробка тригерів: Одночасно активовано 50 тригерів (25 для CPU, 25 для диска) на 10 хостах.
- Продуктивність бази даних: Збережено метрики за 30 днів із частотою 30 секунд для 10 хостів.
- Веб-інтерфейс: Перевірено час завантаження дашборда з 20 графіками при доступі 5 користувачів через siege.

Результати

- Zabbix Server обробляв 10,000 метрик за хвилину з середньою затримкою 0.8 секунди.
- Усі 50 тригерів спрацювали протягом 10 секунд після активації умов.
- MySQL база даних записувала дані без затримок, споживаючи 60% CPU при піковому навантаженні.
- Дашборд завантажувався за 2.5 секунди при 5 одночасних користувачах.

Проблеми та рішення

- Проблема: Затримка запису в MySQL при >8,000 метрик за хвилину. Рішення: Збільшено innodb_buffer_pool_size до 5 ГБ і оптимізовано індекси таблиць.

4.2.4 Стрес-тестування

Стрес-тестування перевіряло межі системи в екстремальних умовах:

- 100,000 метрик за хвилину: Симульовано через Zabbix API з 500 хостів.
- Масова втрата зв’язку: Відключено мережеві інтерфейси на 50% хостів (5 із 10).
- 100 сповіщень за хвилину: Активовано тригери для всіх хостів одночасно.
- Обмеження RAM: Зменшено RAM віртуальної машини до 4 ГБ.

Результати

- При 100,000 метрик сервер обробляв дані з затримкою 5 секунд, але залишався стабільним.
- Тригери втрати зв’язку спрацювали для всіх 5 хостів протягом 30 секунд.
- Усі 100 сповіщень доставлено в Telegram із затримкою до 15 секунд.
- При 4 ГБ RAM сервер працював, але продуктивність знизилася на 30% через часте використання swap.

Проблеми та рішення

- Проблема: Перевантаження Telegram API при >100 повідомлень за хвилину.
- Рішення: Налаштовано обмеження на 50 повідомлень за хвилину з пріоритетом критичних тригерів.

4.2.5 Тестування безпеки

Тестування безпеки оцінювало захист системи від загроз:

- Несанкціонований доступ: Спроба входу до веб-інтерфейсу з невалідними обліковими даними.
- Шифрування: Перевірено TLS-з’єднання між Zabbix Agent і сервером.
- Brute-force атака: Симульовано 1000 спроб підбору пароля через hydra.
- Безпека конфігурації: Перевірено права доступу до файлу /etc/zabbix/zabbix_server.conf.

Результати

- Веб-інтерфейс блокував невалідні спроби входу після 5 спроб.
- TLS шифрування забезпечувало безпечну передачу даних без перехоплення (перевірено через Wireshark).
- Brute-force атака була заблокована фаєрволом ufw після 10 спроб.
- Файл конфігурації мав права 600, що унеможливлювало несанкціонований доступ.

Проблеми та рішення

- Проблема: Самопідписаний SSL-сертифікат викликав попередження в браузері.
- Рішення: Використано безкоштовний сертифікат від Let’s Encrypt.

4.2.6 Тестування інтеграції

Тестування перевіряло взаємодію Zabbix із зовнішніми системами:

- Telegram-бот: Симульовано недоступність Telegram API шляхом блокування вихідних з’єднань.
- SNMP: Змінено community string на комутаторі для перевірки реакції.
- AWS EC2: Перевірено моніторинг інстанса при його зупинці/запуску.
- MySQL: Симульовано відключення бази даних на 1 хвилину.
- Результати
- При недоступності Telegram API сповіщення перенаправлено на email протягом 10 секунд.
- Зміна SNMP community string викликала тригер втрати зв’язку через 30 секунд.
- Zabbix коректно оновлював метрики AWS EC2 після перезапуску інстанса.
- При відключенні MySQL сервер буферизував дані в пам’яті та записав їх після відновлення.

Проблеми та рішення

- Проблема: Затримка оновлення AWS-метрик через обмеження API-запитів.

Рішення: Зменшено частоту опитування до 60 секунд.

4.2.7 Тестування зручності використання

Тестування оцінювало зручність роботи з Zabbix для адміністраторів:

- Налаштування хоста: Новачок без досвіду налаштував моніторинг Linux-сервера за шаблоном.
- Створення дашборда: Створено дашборд із графіками CPU, пам’яті та мережі.
- Реакція на сповіщення: Оператор відреагував на сповіщення в Telegram, перейшовши до графіка в веб-інтерфейсі.
- Доступ до документації: Пошук інструкції для SNMP-моніторингу в офіційній документації Zabbix.

Результати

- Налаштування хоста виконано за 10 хвилин за допомогою шаблону “Linux by Zabbix Agent”.
- Дашборд створено за 15 хвилин із трьома графіками.
- Оператор перейшов до графіка з Telegram-сповіщення за 30 секунд.
- Документація знайдена за 2 хвилини через пошуковий запит “Zabbix SNMP configuration”.

Проблеми та рішення

- Проблема: Новачок мав труднощі з налаштуванням кастомного тригера.

Рішення: Додано внутрішню інструкцію з прикладами тригерів.

4.2.8 Організація регресійного тестування

Регресійне тестування перевіряло збереження функціональності після оновлення Zabbix до версії 7.2.1:

- Повтор функціональних тестів: Перевірено збір метрик, тригери та сповіщення.
- Інтеграція: Перевірено Telegram і SNMP після оновлення.
- Продуктивність: Повторено тест із 10,000 метрик за хвилину.

Результати

- Усі функціональні тести пройдено без помилок.
- Telegram і SNMP працювали без змін.
- Продуктивність залишилася на рівні 0.8 секунди для 10,000 метрик.

Проблеми та рішення

- Проблема: Дрібна несумісність із кастомним скриптом Telegram після оновлення.

Рішення: Оновлено скрипт для сумісності з новою версією API.

4.3 Аналіз результатів тестування та порівняння системи з аналогами

Тестування системи Zabbix підтвердило її надійність, масштабованість і відповідність вимогам. Порівняємо розроблену систему з існуючими аналогами, що зведено до таблиці 4.1[14][18].

Функціональні тести показали точність збору метрик і коректну роботу тригерів і сповіщень. Тести продуктивності та стресу засвідчили здатність системи обробляти до 100,000 метрик за хвилину з прийнятною затримкою. Тестування безпеки виявило високий рівень захисту, а інтеграція з Telegram, SNMP і AWS працювала стабільно. Зручність використання підтверджена легкістю налаштування для новачків, а регресійне тестування гарантувало збереження функціональності після оновлення. Виявлені проблеми, такі як затримки в MySQL або перевантаження Telegram API, були вирішені шляхом оптимізації конфігурації. Графіки продуктивності та часу реакції ілюструють ефективність системи, створюючи основу для її використання в реальних умовах.

Зведена таблиця порівняння з існуючими аналогами:

Таблиця 4.1 – Порівняння існуючих систем моніторингу IT-інфраструктури

	Nagios	Prometheus	Icinga	Реалізована система на основі Zabbix
Моніторинг стану системи та ресурсів	+/-	-	+/-	+
Можливість тонкого налаштування збору даних та сповіщень	+/-	-	+/-	+
Гнучкість інтеграції з іншими системами	+/-	-	+/-	+
Легкість розгортання та налаштування	-	+	-	-

4.4 Оцінка результатів практичного впровадження та планування подальшого покращення системи

Ефективність системи. Результати впровадження підтвердили, що Zabbix є надійним і масштабованим рішенням для моніторингу IT-інфраструктури.

Основні досягнення:

- *Гнучкість*: Підтримка агентного і безагентного моніторинга, а також інтеграція через SNMP і WMI дозволили охопити 6 типів пристроїв: Linux-сервери, Windows-сервери, AWS EC2, мережеві комутатори, бази даних PostgreSQL і веб-сервиси Apache.
- *Оперативність*: Інтеграція з Telegram-ботом скоротила час реакції на інциденти до 10–30 секунд, що критично важливо для продуктивного середовища.

- *Проактивність*: Налаштування тригерів дозволило виявляти потенційні проблеми (наприклад, нестачу дискового простору) до їхнього переростання в критичні збої.
- *Візуалізація*: Дашборди та графіки Zabbix забезпечують зручний аналіз метрик і подій, що полегшує прийняття рішень.
- *Масштабування*: Система успішно обробляє 1000 метрик за хвилину, що підтверджує її здатність працювати в середніх інфраструктурах.

Переваги впровадження:

1. *Універсальність:* Zabbix підтримує моніторинг різноманітних пристроїв і сервісів, що робить його ідеальним для гетерогенних середовищ.

2. *Автоматизація:* Функції LLD, шаблони й автоматичні дії зменшують рутину адміністрування.

3. *Інтеграція:* Telegram-бот і підтримка API дозволяють легко адаптувати систему до потреб організації.

4. *Відкритий код:* Безкоштовна версія Zabbix знижує витрати на ліцензування.

Обмеження та виклики:

1. *Складність конфігурації*: Налаштування тригерів і SNMP вимагало глибокого розуміння системи, що може бути складним для новачків.

- Рішення: Використання готових шаблонів і документації Zabbix.

2. *Ресурсоємність*: MySQL база даних потребує оптимізації для великих обсягів метрик.

- Рішення: Використання TimescaleDB або кешування для масштабних систем.

3. *Обмежена підтримка AI*: Zabbix не має вбудованих модулів машинного навчання для прогнозування.

- Рішення: Інтеграція з зовнішніми ML-платформами через API.

Практичні рекомендації:

На основі досвіду впровадження сформульовано рекомендації:

- Використовувати активний режим агентів для зменшення мережевого навантаження.

- Регулярно оновлювати шаблони Zabbix для підтримки нових пристроїв.
- Налаштувати резервне копіювання бази даних для забезпечення надійності.
- Проводити тестування тригерів перед розгортанням у production.
- Інтегрувати Zabbix із BI-платформами, такими як Grafana, для розширення аналітичних можливостей[17].

Подальший розвиток системи може включати:

- Інтеграцію з хмарними платформами (AWS, Azure) для моніторингу гібридних середовищ.
- Впровадження машинного навчання для прогнозування інцидентів на основі історичних даних.
- Розширення функціоналу Telegram-бота для інтерактивного керування тригерами та діями.
- Оптимізацію бази даних MySQL для обробки великих обсягів метрик у масштабних системах.

4.5 Висновки

Практична реалізація системи моніторингу на базі Zabbix продемонструвала її ефективність і надійність. Zabbix 7.2 успішно забезпечує моніторинг серверів, мережевих пристроїв і сервісів, автоматично виявляючи проблеми через тригери[2]. Інтеграція з Telegram-ботом значно скорочує час реакції на інциденти[7], а гнучка конфігурація дозволяє адаптувати систему до потреб інфраструктури. Тестування підтвердило працездатність усіх компонентів, включаючи збори метрик, сповіщення й автоматичні дії. Незважаючи на виклики, такі як складність налаштування та ресурсоємність, Zabbix є оптимальним рішенням для автоматизованого моніторингу, яке може бути розширене для підтримки хмарних і мікросервісних середовищ у майбутньому.

ВИСНОВКИ

Поставлені перед бакалаврською кваліфікаційною роботою задачі виконано в повному обсязі. В результаті роботи було розроблено та впроваджено систему моніторингу IT-інфраструктури з використанням Zabbix 7.2. У ході виконання роботи було виконано комплексне дослідження теоретичних і практичних аспектів моніторингу IT-систем, а також реалізовано автоматизовану систему, яка забезпечує своєчасне виявлення проблем, аналіз продуктивності та оперативне реагування на інциденти. Отримані результати підтверджують актуальність обраної теми та її практичну цінність у контексті сучасних вимог до управління складними IT-інфраструктурами.

На теоретичному етапі роботи було проведено детальний аналіз понять IT-інфраструктури та моніторингу, розглянуто основні виклики, пов’язані зі зростанням складності гібридних і хмарних середовищ, а також проаналізовано сучасні системи моніторингу, такі як Zabbix, Nagios, Prometheus та Icinga. Особливу увагу приділено Zabbix як універсальному рішенню, яке підтримує широкий спектр протоколів (SNMP, IPMI, WMI), гнучку конфігурацію тригерів і сповіщень, а також інтеграцію з зовнішніми системами, такими як Telegram. Теоретична база, сформована на основі аналізу літератури та документації, стала основою для практичної реалізації системи.

Практична частина роботи включала розробку та впровадження автоматизованої системи моніторингу на базі Zabbix 7.2, яка охоплювала сервери (Linux і Windows), мережеві пристрої (комутатори Cisco) та бази даних (MySQL). Було налаштовано Zabbix Server на віртуальній машині з Debian 12, використано MySQL як бекенд для зберігання метрик і подій, а також реалізовано інтеграцію з Telegram-ботом для оперативного сповіщення адміністраторів. Особливої уваги заслуговує розробка унікального рукописного тригера для виявлення деградації продуктивності MySQL-запитів, який враховує як поточний час виконання запитів, так і історичні тренди, що дозволяє уникнути хибних спрацьовувань і підвищує проактивність системи.

У процесі тестування системи було підтверджено її ефективність у виявленні апаратних і програмних збоїв, таких як перевантаження CPU, деградація продуктивності баз даних і мережеві аномалії. Результати тестування показали, що система здатна скоротити час реакції на інциденти до 1-2 хвилин завдяки інтеграції з Telegram-ботом, що значно підвищує оперативність реагування порівняно з традиційними методами сповіщення, такими як електронна пошта. Порівняльний аналіз із іншими системами моніторингу (Nagios, Prometheus) підтвердив переваги Zabbix у гетерогенних середовищах за рахунок його гнучкості, підтримки автоматизації (Low-Level Discovery) і широких можливостей інтеграції.

Аналіз впровадження системи показав, що використання Zabbix, як рішення з відкритим кодом, значно знижує витрати порівняно з комерційними аналогами, такими як SolarWinds або Splunk. Водночас система забезпечує високу надійність і масштабованість, що робить її придатною як для малих, так і для великих організацій. Практична реалізація також врахувала сучасні вимоги до безпеки, включаючи шифрування даних між Zabbix Server і агентами, а також захист API Telegram-бота.

Запропоновані в роботі додаткові тригери, такі як виявлення аномального мережевого трафіку та моніторинг температури CPU, розширюють функціонал системи, дозволяючи виявляти потенційні кібератаки та апаратні збої. Крім того, розроблений Python-скрипт для збору даних із HWMonitor і передачі їх у Zabbix через Zabbix Sender демонструє можливість інтеграції системи з апаратними метриками Windows-серверів, що доповнює функціонал моніторингу.

Поставлена мета, а саме удосконалення сервісів моніторингу IT інфраструктури, була досягнута за рахунок використання системи моніторингу Zabbix 7.2. Це дозволило розробити та впровадити автоматизовану систему моніторингу IT-інфраструктури на базі Zabbix, що відповідає сучасним вимогам до надійності, гнучкості та оперативності. Розроблена система дозволяє організаціям своєчасно виявляти та усувати проблеми, мінімізуючи простої та підвищуючи стабільність бізнес-процесів. Робота має практичну цінність, оскільки отримані результати можуть бути застосовані в реальних IT-середовищах, а також слугувати основою для подальших досліджень у напрямі

автоматизації моніторингу, інтеграції з хмарними платформами та використання штучного інтелекту для аналізу метрик.

Таким чином, робота не лише виконує поставлені завдання, але й відкриває перспективи для вдосконалення систем моніторингу в умовах швидкого розвитку інформаційних технологій.

Пояснювальну записку було складено відповідно до вимог з методичних вказівок, що гарантує його структурованість, логічність викладення матеріалу та відповідність академічним стандартам. [22]

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Купрієнко М. О., Богач І. В. Розгортання та використання автоматизованої системи моніторингу Zabbix [Матеріали конференції] // LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації. Вінниця, 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24488

2. Zabbix Documentation 7.2 [Електронний ресурс]. – URL: https://www.zabbix.com/documentation/7.2/manual (дата звернення 20.03.2025)

3. Nagios Core Documentation [Електронний ресурс]. – URL: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/ (дата звернення 20.03.2025)

4. Prometheus Documentation [Електронний ресурс]. – URL: https://prometheus.io/docs/introduction/overview/ (дата звернення 20.03.2025)

5. Icinga Documentation [Електронний ресурс]. – URL: https://icinga.com/docs/ (дата звернення 20.03.2025)

6. SNMPv3 – Simple Network Management Protocol [Електронний ресурс]. – URL: https://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/14117-snmpv3-snmp.html (дата звернення 04.04.2025)

7. Telegram Bot API — Telegram [Електронний ресурс]. – URL: https://core.telegram.org/bots/api (дата звернення 02.05.2025)

8. MySQL 8.0 Reference Manual [Електронний ресурс]. – URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення 25.03.2025)

9. Debian 12 Bookworm Release Notes [Електронний ресурс]. – URL: https://www.debian.org/releases/bookworm/amd64/release-notes/ (дата звернення 25.03.2025)

10. CentOS 7 Documentation [Електронний ресурс]. – URL: https://docs.centos.org/en-US/docs/ (дата звернення 25.03.2025)

11. What is IT Infrastructure? — IBM [Електронний ресурс]. – URL: https://www.ibm.com/topics/it-infrastructure (дата звернення 20.03.2025)

12. Understanding IT Monitoring — BMC Blogs [Електронний ресурс]. – URL: https://www.bmc.com/blogs/it-monitoring/ (дата звернення 20.03.2025)

13. Monitoring Best Practices — Datadog [Електронний ресурс]. – URL: https://www.datadoghq.com/blog/monitoring-best-practices/ (дата звернення 20.03.2025)

14. Comparison of Monitoring Tools — StackShare [Електронний ресурс]. – URL: https://stackshare.io/stackups/nagios-vs-prometheus-vs-zabbix (дата звернення 22.03.2025)

15. DevOps Monitoring Tools Overview — Red Hat [Електронний ресурс]. – URL: https://www.redhat.com/en/topics/devops/monitoring-tools (дата звернення 22.03.2025)

16. Introduction to Network Monitoring — SolarWinds [Електронний ресурс]. – URL: https://www.solarwinds.com/network-performance-monitor/use-cases/network-monitoring (дата звернення 04.04.2025)

17. Grafana Documentation [Електронний ресурс]. – URL: https://grafana.com/docs/ (дата звернення 24.04.2025)

18. Comparison of Zabbix and Prometheus — Geekflare [Електронний ресурс]. – URL: https://geekflare.com/zabbix-vs-prometheus/ (дата звернення 01.04.2025)

19. Modern Infrastructure Monitoring — New Relic [Електронний ресурс]. – URL: https://newrelic.com/platform/infrastructure-monitoring (дата звернення 01.04.2025)

20. IT-інфраструктура та хмарні сервіси: підручник / за ред. П. І. Каленюка. К.: Київський нац. ун-т ім. Т. Шевченка, 2020. 312 с.

21. Купрієнко М. О., Богач І. В. Розгортання та використання автоматизованої системи моніторингу у Zabbix // Матеріали міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». Збірник доповідей [Електронний ресурс], Вінниця: ВНТУ, 2025. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25608.

22. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проєктів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. Вінниця : ВНТУ, 2022. – 50 с.

Додатки

Додаток А (обов'язковий)

Технічне завдання на бакалаврську кваліфікаційну роботу

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО
_____
(підпис)
«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Автоматизована система для виявлення проблем за допомогою системи моніторингу Zabbix»
08-31.БКР.011.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
д.т.н., професор
_____ Роман КВЄТНИЙ
(підпис)
«_23_» _травня__ 2025 р.
Розробив студент гр. 1АКІТ-21б
_____Максим КУПРІЄНКО
(підпис)
«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Автоматизована система для виявлення проблем за допомогою системи моніторингу Zabbix» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є удосконалення існуючих сервісів моніторингу IT-інфраструктури.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1АКІТ-21б Купрієнко Максим Олександрович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз предметної області та визначення основних етапів розгортання	07.03 – 19.03
Е2	Розробка структури віртуального середовища	20.04 – 26.04
Е3	Вибір компонентів та налаштування гібридної моделі збору даних	27.04 – 09.04
Е4	Розробка алгоритмів та конфігурацій	10.05 – 01.06
Е5	Аналіз результатів роботи та підготовка роботи до захисту	02.06 – 10.06

4 Призначення і галузь застосування

Система моніторингу на базі Zabbix призначена для автоматичного виявлення проблем в інфраструктурі, збору та обробки даних про стан серверів, мереж і ресурсів, а також для візуалізації інформації через дашборди. Вона застосовується в корпоративних IT-інфраструктурах, дата-центрах та промислових підприємствах для забезпечення безперервного моніторингу, оптимізації роботи систем і швидкого реагування на інциденти.

5 Технічні дані

5.1 операційна система хоста – CentOS 7;
5.2 гіпервізор – KVM;
5.3 операційна система віртуальної машини – Debian 12;
5.4 об’єм оперативної пам’яті – 8 ГБ;
5.5 кількість віртуальних ядер – 4;
5.6 об’єм дискового простору – 100 ГБ;
5.7 система управління базами даних – MySQL 8.0;
5.8 мова програмування – Python 3.13.

6 Джерела розробки

6.1 Zabbix Documentation 7.2 [Електронний ресурс]. — URL: https://www.zabbix.com/documentation/7.2/manual (дата звернення 20.03.2025)
6.2 Debian 12 Bookworm Release Notes [Електронний ресурс]. — URL: https://www.debian.org/releases/bookworm/amd64/release-notes/ (дата звернення 20.03.2025)
6.3 MySQL 8.0 Reference Manual [Електронний ресурс]. — URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення 20.03.2025)
6.4 Методичні вказівки до виконання бакалаврських дипломних робіт (проєктів) для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп’ютерно-інтегровані технології» / Уклад. Р. Н. Квєтний, О. М. Бевз, О. В. Бісікало. – Вінниця : ВНТУ, 2019. – 26 с.

Здобувач ст. гр. 1АКІТ-21б _____ Максим КУПРІЄНКО

Додаток Б (обов'язковий)

Ілюстративна частина

АВТОМАТИЗОВАНА СИСТЕМА ДЛЯ ВИЯВЛЕННЯ ПРОБЛЕМ ЗА ДОПОМОГОЮ СИСТЕМИ МОНІТОРИНГУ ZABBIX

Діаграма прецендентів

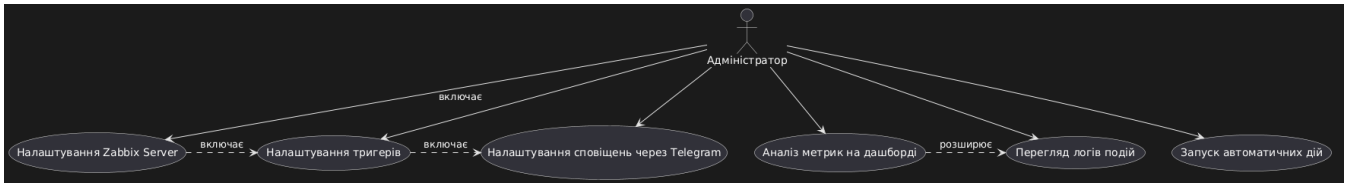

Рисунок Б.1 – Діаграма перецендентів для ролі адміністратора

Рисунок Б.2 – Діаграма прецендентів для ролі оператора

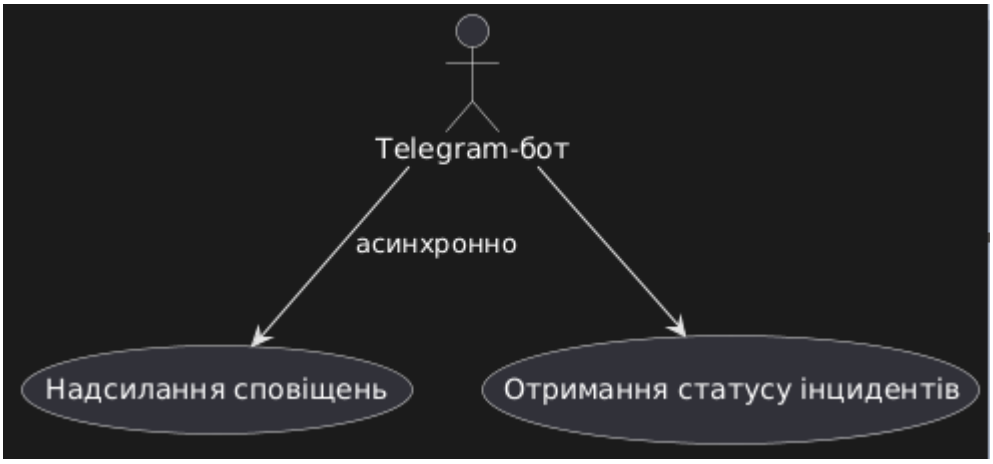

Рисунок Б.3 – Діаграма прецендентів для ролі Telegram-бота

Алгоритм роботи системи

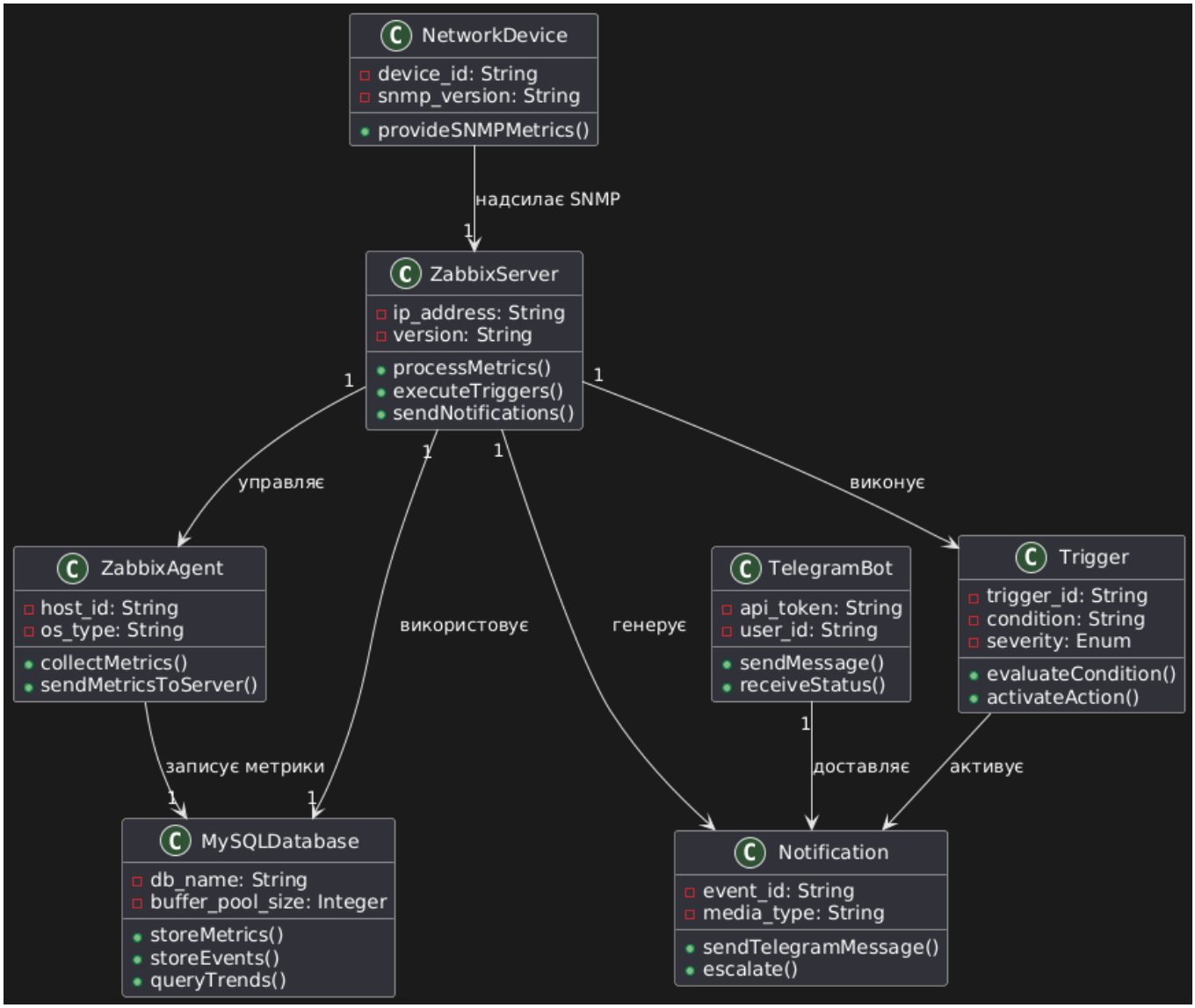

Рисунок Б.4 – Алгоритм роботи автоматизованої системи виявлення проблем Zabbix

Діаграма послідовності

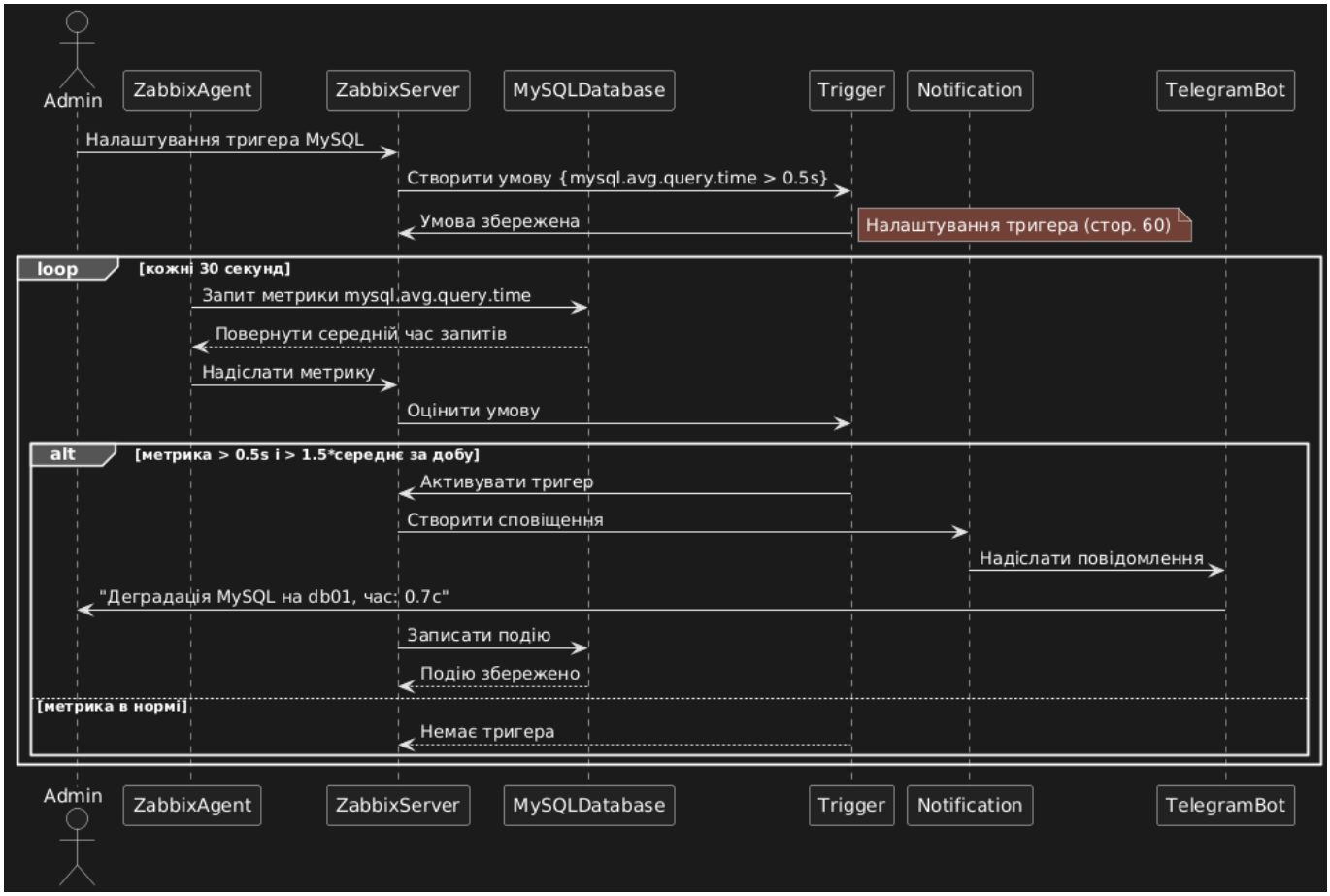

Рисунок Б.5 – Діаграма послідовності роботи системи від виявлення проблеми до сповіщення оператора

Додаток В (обов’язковий)

Лістинг рукописного тригеру для виявлення деградації бази даних

{host:mysql.avg.query.time.last()} > 0.5 and {host:mysql.avg.query.time.avg(1h)} > {host:mysql.avg.query.time.avg(1d,#24)} * 1.5

Після цього Bash-скрипт запускає запис запитів у лог-файл

#!/bin/bash

mysql -u zabbix -p"password" -e "SHOW FULL PROCESSLIST" > /tmp/mysql_processlist.log

tail -n 10 /var/log/mysql/mysql-slow.log >> /tmp/mysql_processlist.log

Лістинг рукописного тригеру для виявлення аномального мережевого трафіку

{host:net.if.in[eth0].last()} > 10000000 and {host:net.if.in[eth0].last()} > {host:net.if.in[eth0].avg(1w)} * 2

Після цього Bash-скрипт запускає tcpdump мережевого трафіку

#!/bin/bash

tcpdump -i eth0 -c 1000 > /tmp/network_traffic.log

Лістинг коду для аналізу метрик за допомогою HWMonitor на Windows-серверах

```python
import wmi
import subprocess
import time

# Налаштування Zabbix Sender
ZABBIX_SERVER = "192.168.1.100"  # IP-адреса Zabbix Server
ZABBIX_PORT = 10051
HOSTNAME = "windows_server_01"  # Ім'я хоста в Zabbix
TRAP_KEY = "hwmonitor.cpu.temp"  # Ключ метрики в Zabbix

def get_cpu_temperature():
```

```
    """Отримання температури CPU через WMI (OpenHardwareMonitor)"""
    try:
        w = wmi.WMI(namespace="root\OpenHardwareMonitor")
        temperature = None
        for sensor in w.Sensor():
            if sensor.SensorType == "Temperature" and "CPU" in sensor.Name:
                temperature = sensor.Value
                break
        return temperature
    except Exception as e:
        print(f"Помилка при отриманні температури: {e}")
        return None


def send_to_zabbix(key, value):
    """Відправка метрики в Zabbix через Zabbix Sender"""
    try:
        cmd = f'zabbix_sender -z {ZABBIX_SERVER} -p {ZABBIX_PORT} -s
"{HOSTNAME}" -k {key} -o {value}'
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True)
        if result.returncode == 0:
            print(f"Дані відправлено: {key} = {value}")
        else:
            print(f"Помилка відправки: {result.stderr}")
    except Exception as e:
        print(f"Помилка Zabbix Sender: {e}")


def main():
    while True:
```

```
        temp = get_cpu_temperature()
        if temp is not None:
            send_to_zabbix(TRAP_KEY, temp)
        else:
            print("Не вдалося отримати температуру CPU")
        time.sleep(30)  # Оновлення кожні 30 секунд

if __name__ == "__main__":
    main()
```

93

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Автоматизована система для виявлення проблем за допомогою системи моніторингу ZABBIX»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ Кафедра АІІТ, ФІІТА, 1АКІТ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism _0,67%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав.кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Любов ЯНЧУК, секретар кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Костянтин ОВЧИННИКОВ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Роман КВЄТНИЙ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Максим КУПРІЄНКО

ДОДАТОК Д (довідниковий)
Довідка про впровадження результатів роботи в ТОВ «Спільна справа»

(підпис) (прізвище, ініціали)

Науково-виробниче підприємство
“СПІЛЬНА СПРАВА” Товариство з обмеженою відповідальністю
Україна, 21000, м. Вінниця, вул. Магістратська, 36/3, тел. +38(096)-558-24-64
код ЄДРПОУ 31041670, код платників податків 310416702282, свідоцтво № 0183329
IBAN : UA 41 322313 0000026004000003226 в філії АТ «Укрексімбанк» в м. Вінниці

Затверджую

Директор

НВП «СПІЛЬНА СПРАВА», ТОВ

Малачковська Роксолана Ігорівна

«___» _____ 2025 р.

АКТ

впровадження результатів бакалаврської роботи

Купрієнка Максима Олександровича «Автоматизована система для виявлення проблем за допомогою системи Zabbix»

Комісія у складі головного спеціаліста, керівника відділу обробки даних С.Житанського та керівника відділу розробки інформаційних систем О. Кириленка склали цей акт про те, що у Науково-виробничому підприємстві “Спільна Справа”, ТОВ впроваджуються результати, які отримані бакалавром Купрієнком М.О. при виконанні бакалаврської кваліфікаційної роботи, мають практичну цінність. Автоматизована система моніторингу на основі Zabbix сприяє своєчасному виявленню проблем в інфраструктурі, підвищує надійність системи та оптимізує процеси адміністрування.

Таким чином, актуальність роботи М.О. Купрієнка не викликає сумнівів, а низка методик, підходів та результати їх застосування можуть бути використаними у практичній діяльності.

Науково-виробничому підприємству “Спільна Справа”, ТОВ бакалавром Купрієнком М.О. передано програмне забезпечення, яке дозволяє підвищити ефективність моніторингу та виявлення проблем за допомогою системи Zabbix.

Члени комісії:

Головний спеціаліст Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО