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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Моніторингові системи у корпоративних мережах»

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

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

	Євгеній ГІТЧЕНКО
(підпис)	*Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-41
Євгеній ГІТЧЕНКО
Ім'я, ПРІЗВИЩЕ
Керівник: Юлія БОРОВА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: Сергій БОНДАР
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
_____
«_____» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гітченко Євгеній Олегович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Моніторингові системи у корпоративних мережах _____

керівник кваліфікаційної роботи Юлія БОРОВА,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56

2. Строк подання кваліфікаційної роботи «5» червня 2025р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; офіційна документація системи Zabbix (архітектура, компоненти, функціонал), мережеві протоколи моніторингу: SNMP, ICMP, TCP, UDP, HTTP, HTTPS, SSH, Telnet, топологія мережі (фізична та логічна), типи мережевих архітектур: одноранговi (P2P), клієнт-серверні, розподілені, протоколи транспортного рівня (TCP, UDP), протокол керування адресами (DHCP), IP-протоколи: IPv4 / IPv6, технологія віртуалізації та контейнеризації Docker
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи моніторингу в корпоративних мережах
2. Архітектура та функціональність Zabbix
3. Огляд альтернативних систем моніторингу
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «24» лютого 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02-24.03.2025	Виконано
2	Обробка матеріалу	23.03-12.04.2025	Виконано
3	Розробка теоретичної частини	12.04-14.04.2025	Виконано
4	Розробка моделі мережі	15.04-16.04.2025	Виконано
5	Висновки за результатами аналізу	17.04-21.04.2025	Виконано
6	Оформлення пояснювальної записки	22.04-01.05.2025	Виконано
7	Розробка презентації	01.05-11.05.2025	Виконано
8	Передзахист	27.05-29.05.2025	Виконано
9	Здача в деканат	02.06-05.06.2025	Виконано

Здобувач вищої освіти _____ Євгеній ГІТЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Юлія БОРОВА
(підпис) (Ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 42 стор., 2 рис., 0 табл., 30 джерел.

Мета роботи – Розгортання та налаштування моніторингової системи Zabbix для корпоративної комп'ютерної мережі.

Об'єкт дослідження – Корпоративна комп'ютерна мережа з централізованим моніторингом її компонентів.

Предмет дослідження – Аналіз системи моніторингу як засобу забезпечення стабільності та керованості IT-інфраструктури на прикладі системи Zabbix.

Короткий зміст роботи: У сучасних умовах більшість бізнес-процесів базується на використанні інформаційних технологій. Надійна робота корпоративної мережі безпосередньо впливає на ефективність роботи підприємства. Збої в роботі серверів, мережевого обладнання чи сервісів можуть призводити до зупинки внутрішніх процесів і навіть до фінансових втрат. Саме тому моніторинг усіх ключових елементів інфраструктури є вкрай важливим.

Zabbix є потужною та універсальною системою моніторингу з відкритим кодом, яка дозволяє в режимі реального часу збирати, аналізувати й візуалізувати дані з мережевих пристроїв, серверів, додатків та інших об'єктів. У даній роботі розглядаються особливості архітектури Zabbix, можливості її адаптації до корпоративного середовища, а також практична реалізація моніторингу умовної мережі підприємства.

Було проаналізовано роль систем моніторингу у забезпеченні безперебійної роботи IT-інфраструктури, описано процес впровадження Zabbix із використанням контейнерних технологій та реалізовано базовий кейс моніторингу серверів та мережевого обладнання. Результати підтвердили ефективність використання Zabbix для виявлення критичних станів, зменшення часу реагування на інциденти

та загального підвищення прозорості в управлінні мережею.

КЛЮЧОВІ СЛОВА: СИСТЕМА МОНІТОРИНГУ, КОРПОРАТИВНА МЕРЕЖА, СПОВІЩЕННЯ, ДАШБОРД, ZABBIX, SNMP, АГЕНТ, ПРОКСІ, ІНФРАСТРУКТУРА, СЕРВЕР, ВІЗУАЛІЗАЦІЯ, АЛЕРТИ, ТРИГЕРИ, АВТОМАТИЗАЦІЯ, КОНТЕЙНЕРИЗАЦІЯ

ЗМІСТ

Зміст
ВСТУП 7
1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ 9
1.1 Поняття та місце моніторингу в корпоративних мережах 9
1.2 Історія розвитку моніторингу мереж 10
1.3 Основні цілі та завдання моніторингу 10
1.4 Класифікація та типи моніторингу 12
1.5 Ключові компоненти системи моніторингу 14
1.6 Основні принципи побудови моніторингу 14
1.7 Метрики та показники моніторингу 16
1.8 Виклики та обмеження побудови системи моніторингу 18
1.9 Сучасні тенденції розвитку моніторингу 20
2 АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІСТЬ ZABBIX 24
2.1 Основні компоненти Zabbix 24
2.2 Взаємодія між компонентами системи 27
2.3. Принципи обробки даних 29
2.4 Автоматизація конфігурації 31
2.5 Засоби візуалізації 32
2.6 Безпека та контроль доступу 34
2.7 Інтеграція з хмарними сервісами моніторингу 36
3 ОГЛЯД АЛЬТЕРНАТИВНИХ СИСТЕМ МОНІТОРИНГУ 41
3.1 Nagios 41
3.2. Icinga 42
3.3. Prometheus 44
3.4. PRTG Network Monitor 46
3.5. SolarWinds Orion 48
3.6. Datadog 49
3.7. Grafana 50
ВИСНОВКИ 52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 54

ВСТУП

У сучасному світі, який стрімко розвивається, і де інформаційні технології відіграють усе важливішу роль у функціонуванні як окремих підприємств, так і цілих галузей, зростає потреба в ефективному контролі за роботою комп'ютерних систем. Необхідність постійного, безперервного нагляду за технічним станом мережевого обладнання, серверів, робочих станцій, а також за різноманітними інформаційними ресурсами стає ключовим аспектом стабільної роботи будь-якої організації. Збої в роботі інформаційних систем можуть призводити до серйозних наслідків – від фінансових втрат до повної зупинки бізнес-процесів. Саме тому питання моніторингу, своєчасного виявлення проблем і оперативного реагування на них постає надзвичайно гостро.

У зв'язку з цим, вивчення систем моніторингу в корпоративних мережах набуває особливої актуальності. Моніторингові системи дозволяють своєчасно виявляти порушення у роботі мережі, знаходити вузькі місця, прогнозувати потенційні проблеми та загалом підвищувати ефективність функціонування інформаційної інфраструктури. Більше того, вони надають адміністраторам інструменти для збору статистичних даних, аналітики та планування масштабування інфраструктури в майбутньому. Таким чином, моніторинг стає не лише інструментом контролю, але й засобом стратегічного управління IT-ресурсами.

Метою цієї курсової роботи є детальне вивчення теоретичних аспектів побудови та функціонування моніторингових систем у корпоративних мережах, аналіз існуючих рішень на ринку, а також практична реалізація моніторингу з використанням однієї з найпоширеніших систем – Zabbix.

Завданнями курсової роботи є:

— проаналізувати основні принципи побудови систем моніторингу;

— дослідити класифікацію моніторингових систем та їх призначення;

— порівняти різні популярні системи, що застосовуються в сучасних мережах;

— детально розглянути архітектуру, функціональні можливості та особливості роботи Zabbix;

— здійснити практичне впровадження моніторингу за допомогою Zabbix в умовній корпоративній мережі;

— зробити висновки щодо ефективності та доцільності використання таких систем.

Об’єктом дослідження виступає процес моніторингу мережевої інфраструктури в межах корпоративної IT-системи. Предметом дослідження є безпосередньо моніторингові системи, їх інструменти, методи та засоби, які забезпечують ефективне функціонування мережі та дозволяють здійснювати контроль за її елементами.

Методи дослідження включають аналіз наукових і технічних джерел, порівняльний аналіз програмного забезпечення, моделювання, тестування, а також практичну реалізацію моніторингу на основі реального або змодельованого середовища.

Таким чином, дана курсова робота спрямована на комплексне вивчення теми моніторингу корпоративних мереж, що є надзвичайно важливим аспектом сучасного адміністрування IT-інфраструктури, й дозволить краще зрозуміти, як саме досягається стабільність, безперебійність і безпека мережевої взаємодії в умовах динамічного цифрового середовища.

1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ

Моніторинг у корпоративних мережах є невід'ємною складовою управління сучасними інформаційними системами. З огляду на стрімке зростання обсягів передаваних даних, ускладнення структур мереж та збільшення кількості критичних бізнес-процесів, від надійності роботи мережевої інфраструктури залежить ефективність роботи підприємства в цілому. У цьому розділі ми детально розглянемо основні поняття, цілі, завдання, функціональні компоненти та принципи побудови моніторингу в корпоративних мережах.

1.1 Поняття та місце моніторингу в корпоративних мережах

Моніторинг (англ. monitoring) – це процес безперервного спостереження за станом об'єктів, систем або процесів із метою отримання оперативної інформації про їхню поточну працездатність, виявлення відхилень від нормального стану та своєчасного реагування на потенційно критичні події. У контексті корпоративних мереж моніторинг охоплює збір і аналіз даних про стан мережевих пристроїв (комутатори, маршрутизатори, міжмережеві екрани тощо), серверів, віртуальних машин, додатків, сервісів та інших компонентів інфраструктури.

Метою моніторингу є:

— Підтримка безперебійності – виявлення й усунення несправностей до того, як вони призведуть до зупинки бізнес-процесів;

— Оптимізація продуктивності – аналіз навантаження на ресурси та їхня оптимізація (наприклад, балансування навантаження, планування масштабування);

— Забезпечення безпеки – виявлення підозрілої активності, аномалій у трафіку, спроб несанкціонованого доступу;

— Планування розвитку – накопичення статистичних даних для прогнозування потреб, планування оновлення обладнання та розширення інфраструктури;

— Звітування й аудит – формування різноманітних звітів для керівництва, демонстрація відповідності стандартам (ISO, GDPR тощо).

Узагальнено, моніторинг в корпоративній мережі спрямований на досягнення стабільності, прозорості й передбачуваності її роботи, що є базовими умовами для успішного функціонування будь-якої організації.

1.2 Історія розвитку моніторингу мереж

Ідея моніторингу мереж бере початок ще з періоду розгортання великих пакетних комутованих мереж у 1970–1980-х роках. Тоді в основному збір даних про стан мереж відбувався за допомогою протоколів нижчих рівнів (наприклад, ICMP – пінгування, SNMP v1). З часом, із поширенням TCP/IP, виникла необхідність більш гнучкого підходу: додали можливість віддаленого збирання статистики, почала використовуватися низка засобів для агрегації та візуалізації даних (RRDTool у середині 2000 х).

Зі збільшенням складності мережевих архітектур та переходом до віртуалізації/хмарних середовищ виникли нові виклики: необхідність моніторингу контейнерів, мікросервісів, хмарних сервісів тощо. З’явилася потреба у єдиних платформах, які поєднують збір SNMP-, агенто-, HTTP-/HTTPS-метрик із їхньою агрегацією, побудовою графіків і формуванням гнучкої системи оповіщення. Саме так розвинулися сучасні системи моніторингу: Zabbix, Prometheus, Nagios, PRTG, SolarWinds, Splunk тощо.

1.3 Основні цілі та завдання моніторингу

Метою моніторингу є не просто отримання показників про завантаження процесора чи обсяг трафіку. Йдеться про інформаційно-аналітичну систему, яка на підставі зібраних даних самостійно визначає аномальні стани й повідомляє про них відповідальних осіб. Деякі з цих цілей наведено нижче

a) Оперативне виявлення відмов і інцидентів

1) Виявлення відмов у роботі обладнання чи сервісів. Наприклад, якщо маршрутизатор перестає відповідати на запити, або служба бази даних не приймає з’єднань, система моніторингу фіксує цю подію та формує повідомлення;

2) Своєчасне керування ресурсами. При зростанні навантаження на сервер до критичних значень відразу відправляється сигнал, що дає можливість перенести частину навантаження або збільшити ресурси (вертикальне чи горизонтальне масштабування).

b) Аналіз продуктивності

1) Збір довготривалих трендів. Наприклад, середня завантаженість CPU за останній місяць, обсяг збережених логів, використання ОЗП тощо.

2) Планування розширення. На основі трендів IT-адміністратор може зробити висновок, що через півроку або рік поточна конфігурація буде недостатньою, й спланувати закупівлю додаткового обладнання чи впровадження нових сервісів.

c) Забезпечення безпеки та відповідності

1) Виявлення аномалій у мережевому трафіку. Наприклад, несподіване збільшення вхідного трафіку може свідчити про DDoS-атаку.

2) Контроль змін конфігурацій. Фіксація того, коли й хто вносив зміни до налаштувань мережевих пристроїв або серверів.

3) Аудит доступу. Логування спроб входу, невдалих авторизацій, виконання привілейованих команд на ключових серверах.

d) Звітність і підзвітність

1) Регулярні звіти для керівництва. Наприклад, щотижневий або

щомісячний звіт про кількість інцидентів, середній час реакції на них, загальну доступність сервісів (SLA).

2) Підзвітність перед аудиторами. У разі проведення внутрішнього або зовнішнього аудиту необхідно надати підтвердження, що система моніторингу працювала, дані збиралися належним чином, й відповідні інциденти були задокументовані.

1.4 Класифікація та типи моніторингу

Моніторингова сфера дуже широка, тому зазвичай виділяють кілька різновидів моніторингу, які можуть поєднуватися або використовуватися окремо залежно від потреб організації.

а) За об’єктами спостереження існує:

1) Мережевий моніторинг - контроль стану мережевих пристроїв (комутатори, маршрутизатори, точки доступу, міжмережеві екрани). Вимірювання параметрів каналу: затримка, втрата пакетів, пропускна здатність;

2) Серверний моніторинг - збирання метрик операційної системи: CPU, ОЗП, дисковий простір, використання I/O. Спостереження за станом процесів та служб (наприклад, служби баз даних, веб-сервери);

3) Моніторинг додатків (Application Performance Monitoring, APM) - відстеження внутрішніх показників програмного забезпечення: час виконання транзакцій, кількість запитів, частота помилок, затримки в обробці. Наприклад, у веб-додатку відстежується час відповіді на запит, кількість сесій, кількість помилок;

4) Моніторинг безпеки - аналіз журналів (SIEM – Security Information and Event Management). Сповіщення про підозрілу активність:

велику кількість невдалих спроб входу, сканування портів, незвичні патерни трафіку.

b) За методами збору даних

1) Агентський моніторинг (Active / Passive Checks). Active checks: агент самініціативно збирає дані та передає їх на сервер (наприклад, Zabbix-агент при активному опитуванні). Passive checks: сервер надсилає запит до агента, а агент відповідає зібраними даними;

2) Безагентний моніторинг (Agentless Monitoring) – використання протоколів SNMP, WMI, SSH для отримання даних без встановлення спеціального агента;

3) Trap-орієнтований моніторинг – пристрій чи програма надсилає повідомлення (trap, snmp trap) на сервер моніторингу при настанні певної події;

4) Web-моніторинг (Synthetics, Web Scenarios) – симуляція дій користувача: перехід за посиланнями, авторизація на сайті, перевірка наявності ключових елементів (наприклад, тексту, коду відповіді);

5) Лог-моніторинг – збирання й аналіз логів із різних джерел: системних журналів Windows/Linux, логів додатків (Apache, Nginx, Tomcat тощо).

c) За спектром охоплення

1) Локальний (infrastructure-level) моніторинг – контроль ресурсів окремих серверів або пристроїв;

2) Глобальний (end-to-end) моніторинг – оцінка кінцевого користувацького досвіду: час завантаження сторінки, відгук API, доступність сервісів з різних географічних точок;

3) Бізнес-орієнтований моніторинг – вимірювання ключових бізнес-метрик: кількість транзакцій, дохід за годину, кількість успішних платежів тощо.

1.5 Ключові компоненти системи моніторингу

Незалежно від конкретного програмного забезпечення (Zabbix, Prometheus, Nagios та ін.), у кожній системі моніторингу можна виділити декілька загальних функціональних блоків:

— *Джерела даних* (Data Sources). Агенти, SNMP-пристрої, web-сценарії, API-інтерфейси, лог-сервери;

— *Сервер збору та обробки даних* (Monitoring Server / Core) – Приймає дані від джерел, аналізує їх відповідно до конфігурованих правил, зберігає у базі даних;

— *База даних* (Time-series Database / SQL-СУБД) – Зберігає отримані метрики, історичні дані, події, конфігурації;

— *Компонент алертингу* (Alerting / Notification Engine) – Формує повідомлення (email, SMS, Telegram, Slack тощо) у разі виявлення аномалій, спрацювання тригерів;

— *Візуалізаційний шар* (Frontend / Dashboard) – Доступ через веб-інтерфейс для перегляду графіків, дашбордів, статусів хостів та сервісів, розподілу навантаження;

— *Інтерфейси програмування* (API / SDK) – Дозволяють автоматизувати конфігурацію, отримувати дані для зовнішніх систем аналітики, інтегрувати моніторинг із DevOps-конвеєрами.

1.6 Основні принципи побудови моніторингу

Реалізація ефективної моніторингової системи ґрунтується на ряді

принципів, які гарантують точність, надійність та корисність інформації для кінцевих користувачів і адміністраторів. Нижче наведено перелік таких принципів з короткими поясненнями.

— Релевантність – дані, що збираються, повинні відповідати бізнес-цілям та очікуванням організації. Зайвий обсяг непотрібної інформації лише ускладнює аналіз і збільшує навантаження на систему;

— Точність й достовірність – методи збору даних мають гарантувати мінімальну похибку. Наприклад, при використанні SNMP-запитів варто звертати увагу на версії протоколу, MIB-файли, коректне відображення OID;

— Своєчасність – затримки в зборі чи обробці метрик можуть призводити до пізнього реагування на критичні події. Для деяких сценаріїв необхідна майже реальна миттєва реакція (наприклад, відмова кластера баз даних викликає негайну необхідність втручання);

— Масштабованість – система моніторингу повинна безболісно «рости» разом із інфраструктурою. Це означає можливість додавання нових хостів, розширення обсягу зберігання даних, використання проксі/агентів для розподілу навантаження;

— Гнучкість і адаптивність – моніторингове рішення має підтримувати як статичні, так і динамічні середовища: фізичні сервери, віртуальні машини, контейнери, хмарні сервіси. Уміння автоматично виявляти нові ресурси та адаптуватися до змін є важливим чинником;

— Автоматизація конфігурації – використання шаблонів, механізмів auto-discovery, API-інтерфейсів для автоматичного додавання хостів, елементів моніторингу, створення дій і сповіщень. Це мінімізує людський фактор і ризики помилок при розгортанні;

— Надійність і стійкість – кожен компонент системи моніторингу (сервер, база даних, агенти) має бути високодоступним. Наприклад, використання кластерного рішення для бази даних, резервних серверів, балансування навантаження;

— Ефективність використання ресурсів – інструменти моніторингу повинні забезпечувати мінімальне навантаження на ті системи, які контролюються. Наприклад, налаштування інтервалів збору даних так, щоб не перевантажувати мережу надто частими запитами;

— Принцип «one metric, one purpose» – кожна метрика має чітко відображати конкретний показник. Використання складних агрегованих метрик із багатьма змінними може знизити прозорість аналізу;

— Доступність і прозорість – користувачі мають легко отримувати інформацію про стан системи через інтуїтивні дашборди та звіти. Важливо, щоб адміністратор або менеджер бачили оповіщення й могли швидко розібратися у причинах проблеми;

— Безпека – дані моніторингу, особливо якщо мова йде про внутрішню інфраструктуру організації, мають передаватися із використанням шифрування (TLS/SSL), а доступ до системи моніторингу – здійснюватися лише уповноваженими користувачами.

1.7 Метрики та показники моніторингу

Кожна система моніторингу покладається на певний набір метрик. Нижче перераховано категорії метрик, які часто використовують у корпоративних мережах:

а) Метрики продуктивності пристроїв

1) Завантаження процесора (CPU utilization) у відсотках;

2) Використання оперативної пам’яті (RAM utilization), swap-пам’ять;

3) Вхідні/вихідні операції диска (Disk I/O) – операції читання/запису;

4) Температура та стан охолодження (важливо для серверних

стійок, мережевих комутаторів).

b) Метрики мережевого трафіку

1) Кількість переданих і отриманих байтів (Bytes In/Out) на інтерфейсі;

2) Кількість пакетів на секунду (Packets Per Second);

3) Процент втрат пакетів (Packet Loss Rate);

4) Затримка (Latency) та час відповіді (Round-Trip Time).

c) Метрики стану сервісів

1) Доступність служби (Service Availability): чи відповідає порт HTTP (80/443), бази даних (3306, 5432 тощо);

2) Час відповіді сервісу (Response Time) – наприклад, час формування сторінки CMS;

3) Кількість активних підключень (Active Connections) до СУБД або до веб-сервера.

d) Метрики додатків (APM-метрики)

1) Пропускна здатність запитів (Transactions Per Second);

2) Середній час обробки запиту (Average Response Time);

3) Частка помилок (Error Rate) – кількість HTTP 5xx/4xx або кількість винятків у коді.

e) Метрики безпеки

1) Кількість невдалих спроб входу (Failed Login Attempts);

2) Підозрілі IP-адреси (Suspicious IPs) із великою кількістю запитів;

3) Наявність /var/log/auth.log записів про спроби підвищення привілеїв.

f) Метрики використання ресурсів у віртуальному середовищі

1) Завантаження CPU/ОЗП/диску віртуальної машини (VM);

2) Кількість «упаковок» контейнерів (Docker containers) або Kubernetes Pods;

3) Використання ресурсів у контейнерному середовищі: CPU shares, memory limits та ін.

1.8 Виклики та обмеження побудови системи моніторингу

Впровадження й експлуатація повноцінної системи моніторингу в корпоративній мережі стикається з низкою серйозних викликів, які часом є непомітними на перших етапах розгортання, але згодом можуть стати критичними для стабільності та продуктивності всієї інфраструктури. Найперше, це стосується зростання навантаження на систему моніторингу. У невеликій мережі початкові тести можуть справно працювати навіть на середньостатистичному сервері, але в міру того як до моніторингу додається все більше хостів – десятки або сотні віртуальних або фізичних серверів, комутаторів, маршрутизаторів, контейнерів і хмарних сервісів – обсяг даних, які треба збирати, обробляти й зберігати, зростає експоненціально. Відповідно, якщо заздалегідь не передбачити масштабування архітектури моніторингу, він може стати вузьким місцем: сервер збору даних почне «падати» або затримувати оновлення, база даних заповниться, а затримка в обробці метрик призведе до того, що критичні події залишаться непоміченими вчасно.

Ще одним суттєвим обмеженням є різнорідність середовища, яке необхідно контролювати. Сучасні корпоративні мережі часто містять разом як фізичні сервери під керуванням різних дистрибутивів Linux або Windows Server, так і віртуальні машини різних платформ (VMware, Hyper-V, KVM), контейнери (Docker, Kubernetes) та хмарні сервіси (AWS, Azure, GCP). Кожен із цих елементів потребує власного підходу до збору метрик: для Linux-серверів це може бути встановлений агент Zabbix, а для Windows-серверів – окремий агент Windows, у той час як у Kubernetes зазвичай використовують специфічні «експортери» метрик для Prometheus або Zabbix Agent 2 з плагінами. Відповідно, адміністратори повинні знати тонкощі кожної платформи: які саме пакети встановлювати, як налаштувати брандмауери, які OID-галузі у SNMP використовувати, які плагіни підключати тощо. Через це потреба підтримувати актуальні версії агентів, експортерів та їхніх

конфігурацій часто призводить до великих витрат часу на управління середовищем.

Крім того, у закритих чи обмежених середовищах (наприклад, у сегменті мережі, який відділений DMZ або окремим міжмережевим екраном) можуть бути заборонені певні вихідні з'єднання. Наприклад, SNMP-запити можуть бути заблоковані політиками безпеки, а запуск агента Zabbix Active Check може бути неможливим через обмеження портів. У таких випадках доводиться впроваджувати проміжні компоненти (Zabbix Proxy) або переважати на вибір агентського (Active) чи серверного (Passive) методу збору, що може заплутати налаштування і вимагає додаткового їхнього тестування.

Не менш важлива проблема – затримки та похибки при складанні метрик. Навіть якщо мережа має хорошу пропускну здатність, у пікові моменти навантаження або через неповне оновлення топології маршрутизатор може відповідати на SNMP-запити із затримкою, або взагалі не відповідати. Це може спотворити реальний стан системи. Наприклад, якщо використовуються інтервали збирання даних у 30 секунд, але через затримки обробки на сервері або мережеві затримки фактичні інтервали можуть зрости до кількох хвилин – у цей час при різкому стрибку навантаження система зможе повідомити про проблему із запізненням. Для мінімізації таких похибок необхідно ретельно налаштовувати таймаути запитів, кількість повторних спроб, а також робити крок агрегації (rollup) даних на боці агента або проксі, щоб згладити коливання.

Ще одним викликом є потреба забезпечити високий рівень безпеки даних моніторингу. Оскільки моніторинг збирає чутливі показники: від детальної статистики завантаження ресурсів до журналів входу, слід переконатися, що дані передаються у зашифрованому вигляді (TLS/SSL) та доступ до самої системи моніторингу захищений багаторівневими правами доступу. Якщо ж адміністратори економлять час і налаштовують моніторинг із відкритими незашифрованими каналами або спільним обліковим записом адміністратора, це створює уразливість: зловмисник, який отримає доступ до системи моніторингу, зможе бачити всю архітектуру мережі, у тому числі слабкі місця, і найчастіше може навіть зупинити сповіщення, залишивши компанію «в темряві» під час реальної атаки.

Важливою проблемою є також обробка тривожних повідомлень. Невірно встановлені пороги (thresholds) або некоректно налаштовані тригери можуть породжувати масу фальшивих алертів. Наприклад, якщо для CPU встановити поріг у 70 %, а сервер у піковий час стабільно завантажується до 80 %, щохвилини надходитиме повідомлення про перевантаження. Адміністратор почне ігнорувати всі сповіщення, оскільки вважає, що це «черговий збій моніторингу», – замість того, щоб вчасно реагувати на реальні події. Тому необхідно використовувати адаптивні пороги, враховувати середньодобові тренди або застосовувати механізми «затримки» (наприклад, спрацювання тригера лише після того, як показник тримався вище порогу протягом певного часу).

Нарешті, архівування та зберігання довготривалих даних – окрема велика тема. У багатьох компаніях погоджуються зберігати метрики тільки за останні 30 днів, адже далі обсяг бази даних починає критично збільшуватися, що уповільнює відповіді на запити до бази. Проте іноді необхідно мати доступ до статистики за попередні роки, щоб, наприклад, проаналізувати, як змінилася навантаженість після запуску нового сервісу. У такому разі доводиться облаштовувати окремі сховища архівних даних, зробити денормалізацію, використовувати зовнішні сховища на базі Hadoop чи Elastic Stack, окремі time-series бази (InfluxDB, TimescaleDB) і навіть впроваджувати процедури висновку (data retention), щоби не перевантажувати основну систему моніторингу.

Таким чином, кожен із зазначених викликів треба проаналізувати під час проєктування моніторингу, обравши правильну архітектуру, налаштувавши безпеку та оптимальні інтервали збору даних, передбачивши можливість масштабування й коректної обробки великих обсягів інформації.

1.9 Сучасні тенденції розвитку моніторингу

Упродовж останнього десятиліття галузь моніторингу інтенсивно трансформується під впливом двох головних факторів: стрімкого зростання обсягів

даних, які необхідно збирати й аналізувати, та переходу до хмарних і контейнеризованих середовищ. Ось основні тенденції, які сьогодні визначають розвиток систем моніторингу:

Першою і найбільш виразною тенденцією стало поява моделі «Metrics as Code» та активне використання принципів Infrastructure as Code (IaC). У класичному підході конфігурація моніторингу (опис хостів, елементів моніторингу, тригерів, дій) майже завжди здійснювалася вручну через веб-інтерфейс. Проте у великих організаціях це призводило до низької відтворюваності та ризику «людських помилок». Сьогодні розробники дедалі частіше зберігають файли конфігурації моніторингу (YAML, JSON, Terraform- або Ansible-скрипти) у системах контролю версій (Git). Це означає, що кожна зміна (наприклад, новий хост або оновлений шаблон) описується одним рядком у репозиторії, а CI/CD-конвеєри автоматично виконують застосування цих змін. Таким чином, у разі помилки можна швидко відкотитися до попередньої версії, і конфігурація моніторингу стає частиною єдиної автоматизації інфраструктури.

Другою важливою тенденцією є зростання ролі контейнерних оркестраторів і мікросервісної архітектури. Системи на кшталт Kubernetes стали стандартом у багатьох компаніях. Через це потреба у моніторингу окремих контейнерів (Pod), розподілених служб, Namespace та ReplicaSet зросла в рази. Замість одного-двох хостів у піддомені можуть одночасно працювати сотні контейнерів, які можуть постійно з’являтися й зникати. Для їхнього моніторингу використовують експортери метрик (Prometheus Exporter для Zabbix чи Prometheus), спеціальні плагіни, а також сервіси для автоматичного виявлення (service discovery). Завдяки цьому система моніторингу самостійно знаходить нові Pod-и чи сервіси й починає за ними стежити без втручання адміністратора.

Третя тенденція – це інтеграція з AIOps і використання методів машинного навчання для аналізу великих обсягів метрик. Катастрофічне зростання даних у великих підприємствах унеможливило ручний аналіз графіків і трендів. Тому вбудовані алгоритми аномалійного виявлення (anomaly detection) дозволяють у автоматичному режимі відсіювати «шум» коливань і виявляти нетипові патерни,

які можуть свідчити про DDoS-атаку, витік пам’яті або невдалу зміну конфігурації. Деякі сервіси навіть використовують техніки прогнозування (predictive analytics), щоби передбачити, коли навантаження на сервер наблизиться до критичної межі, – і попереджати адміністраторів заздалегідь.

Четверта актуальна тенденція – мультиклаудовість. Багато компаній вже не утримують всю інфраструктуру в одному дата-центрі, а розгортають її одночасно в приватному і публічному хмарних середовищах (AWS розгортає бази даних у RDS, Azure – хостить віртуальні машини, Google Cloud обробляє великі дані). Сучасний моніторинг має забезпечувати уніфікований погляд на ресурси, незалежно від того, чи це фізичний сервер, віртуальна машина в GCP, контейнер у Kubernetes, чи функція AWS Lambda. Для цього використовують хмарні API, інструменти агрегації метрик (CloudWatch, Stackdriver, Azure Monitor), які потім інтегрують у єдину панель керування (Grafana, Zabbix).

П’ята тенденція – удосконалені методи візуалізації. Класичні графіки CPU/RAM у вигляді прямих ліній вже не задовольняють потреб сучасних команд, які хочуть моментально побачити аномалії. Використовують інтерактивні дашборди з можливістю зумування в часі, подвійних осей (дві метрики одночасно), heatmap, treemap, а також інтегровані картографічні відображення (наприклад, фізичні топології дата-центру чи логічні карти залежностей мікросервісів). Завдяки цьому адміністратор може одним поглядом оцінити ситуацію та швидко побачити „чорно-червоні“ ділянки, які потребують негайної уваги.

Нарешті, з розвитком безсерверних (serverless) технологій в API-орієнтованих середовищах з’явилася потреба в моніторингу FaaS (Functions as a Service). Наприклад, AWS Lambda, Google Cloud Functions, Azure Functions виконують малі фрагменти коду «на вимогу», і традиційні методи моніторингу VM-метрик тут не підходять. Моніторинг serverless-функцій фокусується на часі виконання, кількості викликів, помилках, холодному старті, викликах сторонніх API. Дані з CloudWatch чи Stackdriver збирають через відповідні інтеграції й відображають у загальній системі моніторингу, що дозволяє мати єдину картинку для всього стеку (серверів, контейнерів, безсерверних функцій та SaaS-сервісів).

Таким чином, сучасні тенденції свідчать про те, що моніторинг розвивається у напрямку інтегрованості змін у DevOps/DevSecOps-процеси, автоматизації, глибшої аналітики та підтримки будь-яких платформ, від класичних серверів до хмарних безсерверних середовищ.

2 АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІСТЬ ZABBIX

У цьому розділі детально розглядаються складові архітектури Zabbix, принципи взаємодії її компонентів, механізми обробки даних, можливості автоматизації конфігурації, доступні засоби візуалізації та питання забезпечення безпеки й контролю доступу. Використовуючи наданий матеріал, ми значно розширимо кожний підпункт, щоб надати повноцінне уявлення про можливості та особливості системи.

2.1 Основні компоненти Zabbix

Zabbix – це модульна система моніторингу з відкритим кодом, усі елементи якої спроектовані таким чином, щоб забезпечити високу гнучкість, масштабованість та надійність. Кожний із базових компонентів виконує конкретну функцію, і лише в тандемі вони утворюють єдину інфраструктуру моніторингу, здатну працювати як у невеликих локальних середовищах, так і в глобально розподілених системах. Приклад такої інфраструктури наведено на рис. 2.1

Zabbix Server. Серце всієї системи, яке відповідає за збір, обробку та збереження даних моніторингу. Сервер ініціює з’єднання з агентами, проксі та іншими джерелами даних (наприклад, SNMP-пристрої або веб-сценарії), отримує від них значення метрик, аналізує ці значення відповідно до налаштованих тригерів і, у разі необхідності, формує події. Зазначений процес подій може запускати заздалегідь налаштовані дії, такі як надсилання сповіщень (email, SMS, Telegram тощо) або виконання вбудованих сценаріїв. Саме за безперебійну роботу Zabbix Server відповідає вся система моніторингу, оскільки відмови на цьому рівні майже завжди спричиняють втрату контрольованої інформації.

База даних (Database). Усю конфігурацію системи, історичні значення зібраних метрик, параметри елементів, тригерів, дій та інформацію про користувачів зберігається у базі даних. Zabbix офіційно підтримує декілька популярних СКБД: MySQL (та MariaDB), PostgreSQL, SQLite, а також Oracle (хоча

остання використовується значно рідше через ліцензійні обмеження). Для невеликих середовищ може бути достатньо використання SQLite як у вигляді простого файлу, але для масштабніших проєктів практично завжди обирають MySQL або PostgreSQL, бо вони забезпечують кращу продуктивність й опрацьовують великий обсяг даних (історію, тренди) більш оптимально. Важливо правильно налаштувати параметри сховища (параметри кешування, розмір буферів, зони пам'яті), аби уникнути «пробок» при високочастотному зборі метрик.

Веб-інтерфейс (Frontend). Реалізований на PHP та доступний через будь-який сучасний браузер, він забезпечує адміністрування та візуалізацію всіх об'єктів Zabbix. Через веб-інтерфейс адміністратор створює та налаштовує хости, додає шаблони, налаштовує тригери, створює дашборди, переглядає графіки та звіти. Усі дії користувача обробляються через веб-сервер (nginx або Apache з PHP-FPM), що дозволяє забезпечити одночасний доступ кількох адміністраторів. Інтерфейс підтримує багатомовність і надає безліч готових віджетів (графіки, табличні дані, карти, слайди), а також дозволяє створювати власні шаблонізовані панелі для різних категорій користувачів.

Zabbix Proxy. Проксі-компонент призначений для прискорення та оптимізації моніторингу в розподілених середовищах. Проксі може бути розгорнутий у віддалених офісах або філіях, де є обмежене або нестабільне з'єднання з центральним сервером. Замість того, щоб кожний агент напряму надсилав дані до Zabbix Server, дані спочатку передаються на проксі, який локально їх зберігає у власній SQLite- чи MySQL-базі даних, а потім пакетно синхронізує їх із головним сервером. Це значно зменшує навантаження на канал зв'язку та дає можливість продовжувати збирання метрик під час тимчасової відсутності з'єднання з центральним сервером. Проксі також може виконувати роль розвантажувача у сценаріях, коли кількість агентів перевищує можливості єдиного сервера збирання даних.

Zabbix Agent / Agent 2. Агент – це клієнтська програма, що встановлюється на хостах (фізичних або віртуальних машинах) для збору інформації про локальні ресурси: CPU, пам'ять, диски, мережеві інтерфейси, процеси, сервіси тощо. Існує

дві версії: класичний Zabbix Agent і новіший Zabbix Agent 2, реалізований на Go, який пропонує покращену продуктивність, багатопоточність та розширюваність через плагіни. Agent підтримує як «active checks» (коли агент ініціює з'єднання із сервером і надсилає кешовані дані), так і «passive checks» (коли сервер запитує дані напряму у агента). Багато параметрів конфігурації агента, наприклад Server, ServerActive, Hostname, StartAgents, Timeout, визначають, як і куди агент надсилатиме інформацію. За допомогою Zabbix Agent 2 також можна підключати додаткові плагіни, які дозволяють моніторити нестандартні сервіси, наприклад, Redis, Docker, Kubernetes, SNMP через зовнішні експортери та ін.

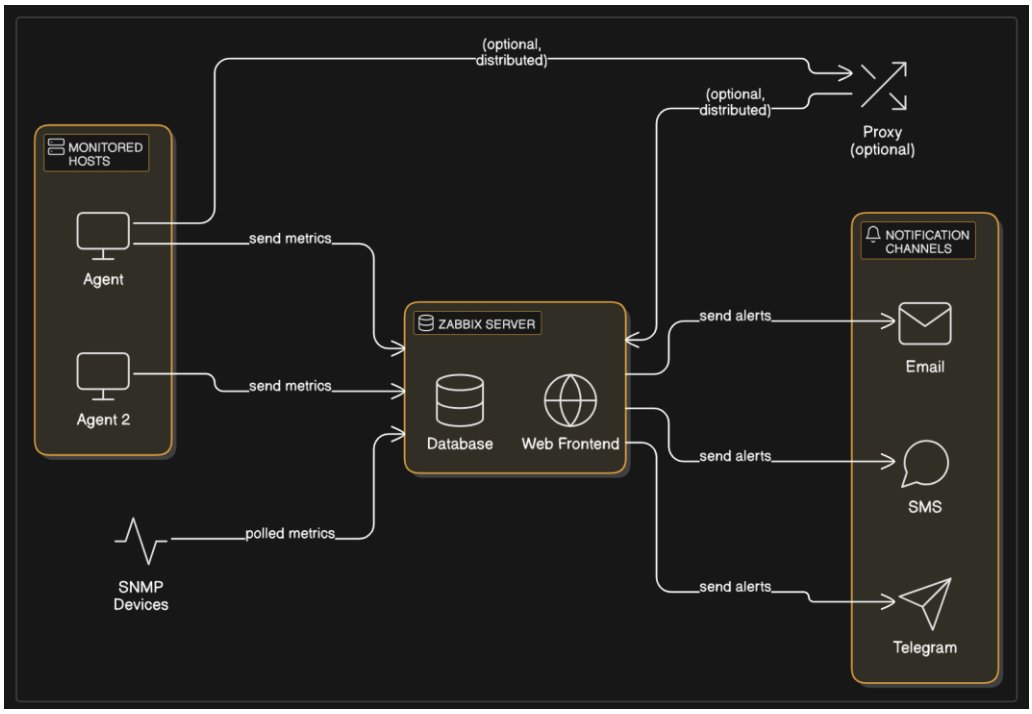


Рисунок 2.1 – Загальна схема роботи Zabbix

Детальніший опис конфігурації агента

а) Файл конфігурації агента зазвичай розташований у /etc/zabbix/zabbix_agentd.conf або у випадку Zabbix Agent 2 – /etc/zabbix/zabbix_agent2.conf. Найважливіші параметри:

1) Server=<IP_адреса_сервера> – IP або DOMEN серверу Zabbix, до якого агент відправляє пасивні перевірки;

2) ServerActive=<IP_адреса_сервера> – IP або DOMEN серверу

Zabbix, до якого агент надсилає активні перевірки (коли агент ініціює зв’язок);

3) Hostname=<ім’я_хоста> – імя хоста в Zabbix, яке має збігатися з зареєстрованим у веб-інтерфейсі;

4) Include=/etc/zabbix/zabbix_agentd.d/*.conf – опціональний параметр для підключення додаткових файлів конфігурації, що дозволяє розбивати налаштування на модулі.

b) Zabbix Agent 2 підтримує додаткові параметри, наприклад:

1) ControlSocket – шлях до Unix-сокета для runtime-керування;

2) EnablePersistentBuffer, PersistentBufferFile, PersistentBufferPeriod – налаштовують локальне кешування даних на боці агента для мінімізації втрат при тимчасових збоях;

3) ForceActiveChecksOnStart – дозволяє агента виконати активні перевірки одразу після перезапуску, а не чекати випадкового інтервалу, що корисно для пришвидшення процесу моніторингу після рестарту;

4) Plugins.<Plugin name>.<Parameter> – конфігурація окремих плагінів (наприклад, Plugins.Docker.Capacity=200).

2.2 Взаємодія між компонентами системи

Коли ми говоримо про взаємодію компонентів Zabbix, маємо на увазі сукупність процесів, які забезпечують безперервний обмін даними між агентами, проксі та сервером, а також консолідування й обробку цих даних в єдиній архітектурі. У класичному сценарії все виглядає таким чином:

a) Джерела даних. До них належать:

1) Zabbix Agent / Agent 2, що встановлюється безпосередньо на хості (сервері, віртуальній машині), дотримуючись правил безпеки (враховуючи, що агент відкриває порт 10050/10051);

2) SNMP-пристрої (мережеві комутатори, маршрутизатори,

міжмережеві екрани), які відповідають на SNMP-запити на порті 161.

3) IPMI для моніторингу апаратного стану серверів на рівні BMC4

b) Веб-сценарії (web scenarios), коли Zabbix Server сам ініціює HTTP/HTTPS-запити до веб-сервісів та аналізує час відповіді або статусний код;

c) Лог-моніторинг, коли Zabbix Server або Proxy «слухають» журнали застосунків (через Zabbix Agent або через external script), що дозволяє виявляти текстові патерни (наприклад, слова «ERROR», «CRITICAL»);

d) Zabbix Agent / Agent 2 (або інші джерела) передають метрики на Zabbix Server:

1) При passive checks сервер виконує запити до агента за допомогою TCP-зібраних даних. Такий підхід зручний для узгодження, коли сервер контролює частоту та обсяг даних;

2) При active checks агент сам ініціює з’єднання з сервером і передає дані пакетами, що оптимізує навантаження на сервер у разі великої кількості агентів;

e) Zabbix Proxy (за наявності) виступає посередником:

1) Агенти підключаються до проксі, передаючи дані, вони кешуються у локальній базі проксі;

2) Проксі періодично надсилає агреговані дані на Zabbix Server через захищений канал.

Якщо підключення до центрального серверу тимчасово відсутнє, проксі продовжує накопичувати дані та надішле їх, коли з’єднання відновиться .

Zabbix Server отримує дані, зберігає їх у базі даних, виконує обчислення тригерів (визначає, чи є певна метрика аномальною або поза встановленими порогами), формує події та запускає дії (сповіщення, виконання зовнішніх скриптів, виклики API інших систем).

Веб-інтерфейс (Frontend) запитує у сервера результати обробки:

- Із бази даних дістає історичні дані для побудови графіків і трендів;
- Відображає список подій, проблем і їхній поточний статус;
- Дозволяє адміністратору вносити зміни в конфігурацію (додавати

хости, шаблони, змінювати налаштування тригерів, налаштовувати права доступу).

— API. Крім веб-інтерфейсу, Zabbix надає REST API, яке дозволяє автоматизувати більшість операцій (створення хостів, налаштування шаблонів, запуск внутрішніх команд). Так, у CI/CD конвеєрах можна додатково створювати нові хости автоматично при розгортанні сервісів, а потім видаляти їх перед зниженням навантаження. Це суттєво спрощує роботу DevOps-команд у сучасних середовищах;

— Завдяки такій архітектурі Zabbix забезпечує високий ступінь гнучкості та можливість масштабування – від невеликих «ланок» моніторингу на 10–20 машин до великих середовищ, що обробляють десятки тисяч хостів.

2.3. Принципи обробки даних

Працюючи в режимі постійного збору, Zabbix керується чітко визначеною послідовністю дій, яка включає реєстрацію хостів, налаштування елементів (items), побудову тригерів та формування дій. Усе це дає змогу перетворити сирі цифрові значення в корисні повідомлення і звіти.

Реєстрація хоста. Перший крок полягає в тому, щоб у веб-інтерфейсі було додано новий хост із певним ім’ям (Hostname) та визначено, яким чином він зв’язуватиметься із сервером (через агента чи proxy). Одночасно вказується, до якої групи хостів належить новий об’єкт (наприклад, “Web Servers”, “Database Servers” тощо).

Створення елементів (items). Елемент (item) – це джерело даних: запит SNMP, ключ Zabbix Agent, запит до API сервісу або web-сценарій. Для кожного елемента вказується:

— ключ (item key), наприклад, system.cpu.load або vfs.fs.size[/,free];

— інтервал оновлення (Update Interval), наприклад, 60 с або 30 с;

— тип перевірки (Active, Passive, SNMP, IPMI, JMX, HTTP);

— тип даних (ціле число, дробове, рядок, лог);

— історія зберігання (скільки днів зберігати детальні (history) дані та тренди (trends)).

Ці налаштування визначають, як часто система витягуватиме значення й куди вони потраплять у базі даних (історія або тренд) .

Налаштування тригерів. Тригери – це логічні вирази, які порівнюють останнє або середнє значення зібраного параметра з певним порогом. Наприклад, вираз

{Template OS Linux:system.cpu.load[percpu,avg5].last()}>5

вважає завантаження CPU за п’ятихвилинним інтервалом критичним, якщо воно перевищує значення 5. Коли тригер переходить із стану «ОК» у стан «Проблема», система створює подію (event) і починає виконувати налаштовані дії (actions). Якщо значення повертається в норму, тригер стає «ОК», адже вираз більше не виконується .

Формування дій (actions). Після спрацьовування тригера Zabbix створює подію, яку можна використовувати як умову для дії. Наприклад, у разі критичного стану CPU автоматично відправляється email директорам відділів і адміністраторам. Можна також задавати складні сценарії ескалації (escalation), коли спочатку надсилається повідомлення одному співробітнику, а в разі відсутності реакції протягом заданого часу – наступному тощо. Операції можуть включати не лише розсилку сповіщень, але й виконання віддалених команд на хості (remote commands), наприклад, перезапуск служби, виконання скрипта діагностики тощо .

Зберігання та агрегація даних. Оскільки історичні дані можуть накопичуватися дуже швидко, Zabbix використовує два основні сховища у базі даних:

– history – зберігає значення метрик із високою роздільною здатністю (залежно від налаштувань, інтервал може бути від кількох секунд до хвилин);

– trends – агрегує дані за довші періоди (наприклад, мінімальні, максимальні та середні значення за годину чи день), що економить простір у базі, але дає змогу бачити довготермінові тренди.

Візуалізація. Усі зібрані дані (як історичні, так і трендові) можуть бути

представлені у вигляді графіків, гістограм, карт, панелей дашбордів. Кожен графік для метрики можна налаштувати як простий лінійний, так і комбінований (наприклад, CPU і пам’ять на одній діаграмі). Карти мережі (Network Maps) дають можливість візуалізувати стан хостів у вигляді географічної карти чи умовної топології, змінюючи кольори вузлів залежно від рівня критичності.

Завдяки такому механізму обробки даних Zabbix здатний реагувати практично миттєво на події в середовищі, а також вести історичний аналіз, необхідний для планування й оптимізації.

2.4 Автоматизація конфігурації

Для зменшення ручної роботи при налаштуванні великої кількості хостів і шаблонів у Zabbix використовується кілька механізмів автоматизації, які допомагають швидко та послідовно вводити зміни в конфігурацію:

Шаблони (Templates). Шаблон – це логічне об’єднання набору елементів (items), тригерів (triggers), графіків (graphs), низькорівневих правил виявлення (LLD, Low-Level Discovery) та дій (actions). Наприклад, шаблон «Template OS Linux» вже містить набір item key, таких як system.cpu.load, system.memory.usage, vfs.fs.size[/,free] тощо, а також попередньо налаштовані тригери, наприклад system.cpu.load[percpu,avg5] > 5. Щоби під’єднати шаблон до нового хосту, достатньо вказати його в налаштуваннях хоста, і всі відповідні елементи почнуть автоматично діставати дані, без потреби налаштовувати кожен item вручну

Автоматичне виявлення (Discovery). Завдяки LLD-механізму Zabbix може автоматично виявляти файлові системи, мережеві інтерфейси, стани процесів або навіть віддалені підмережі через SNMP. Наприклад, якщо на сервері додається новий диск, Zabbix автоматично «побачить» його під час виконання правила виявлення vfs.fs.discovery[/] і створить item, що відслідковуватиме вільний простір диску. Правила виявлення можна налаштовувати як для локальних хостів, так і для мережевих пристроїв, що значно скорочує час налаштування великих середовищ

Agent autoregistration. Якщо агент налаштовано для активних перевірок, він може автоматично створювати хост у Zabbix Server під час першого запуску. Достатньо вказати у конфігурації агента параметр HostMetadata=<some_metadata>, а в інтерфейсі Zabbix створити правило авто реєстрації за метаданими. Наприклад, якщо HostMetadata=linux-web, то всі агенти з цими метаданими будуть автоматично додані до групи «Web Servers» із шаблоном «Template OS Linux Web». Це дуже зручно в масштабних середовищах, коли нові сервери додаються автоматично через IaC (Terraform, Ansible) і лише після остаточної конфігурації агента повинні з'явитися в моніторингу

API. Через REST API Zabbix дозволяє програмно створювати, оновлювати та видаляти майже всі об'єкти: хости, шаблони, item, тригери, карти, дашборди тощо. Це дає змогу інтегрувати моніторинг у CI/CD конвеєри: при кожному новому релізі інфраструктури скрипт на Python чи Bash може додавати або змінювати конфігурацію Zabbix, наприклад, замінювати старі IP-адреси, оновлювати список перевірок або змінювати пороги тригерів залежно від версії ПЗ

Таким чином, поєднання шаблонів, механізмів авто виявлення, агентської авто реєстрації та програмного доступу через API забезпечує значний рівень автоматизації, особливо в середовищах із численними та змінними ресурсами.

2.5 Засоби візуалізації

Однією з найважливіших складових систем моніторингу є можливість наочно та наочним чином представити зібрані дані таким чином, щоб адміністратор міг швидко оцінити поточний стан інфраструктури та виявити проблемні ділянки.

а) *Графіки* (Graphs). Zabbix дозволяє будувати як прості лінійні графіки (одна метрика протягом часу), так і складені (декілька метрик на одному полотні з різними осями). Наприклад, можна одночасно відобразити навантаження CPU та використання пам'яті, що допомагає виявити кореляцію між зростанням CPU та падінням доступного RAM. Також доступний тип «Stacked Graph», коли показники

накладаються одне на одне, що допомагає зрозуміти загальне навантаження на систему;

b) *Дашборди* (Dashboards). Інтерфейс дозволяє створювати кастомізовані дашборди, куди можна додавати різноманітні віджети (widgets):

1) Latest Data – показує останні значення всіх обраних елементів у табличному вигляді;

2) Data Overview – дозволяє швидко переглянути стан усіх хостів у групі за обраними показниками;

3) Network Map – відображає топологію мережі у вигляді графічної карти, де кожен вузол змінює колір залежно від стану (OK, Warning, High, Disaster);

4) Triggers Overview – список усіх тригерів, що знаходяться у статусі «Problem»;

5) Simple Graph – невеликий графічний віджет для відображення однієї метрики;

6) Користувач може вільно переміщувати віджети по полотну, змінювати їх розмір та налаштувати фільтри (наприклад, відображати лише ті хости, що належать до групи «Production Servers»).

c) *Карти мережі* (Network Maps). Карти дозволяють побудувати візуальну модель мережевих зв’язків: у вигляді фізичної топології (узгодженої з документацією) або умовної (коли вузли розташовані для зручності). Кожна лінія зв’язку може містити додаткову інформацію про навантаження на канал (через width/колір лінії). Налаштувавши правильні тригери, можна додавати автоматичні індикатори стану (зелений, жовтий, червоний) до вузлів, що дозволить адміністратору одним поглядом побачити, в якому сегменті мережі виникла проблема

d) *Слайди* (Slideshows). Zabbix підтримує можливість створення слайдів із різних графіків та панелей, які автоматично переглядаються через заданий інтервал часу (наприклад, кожні 10 секунд). Це корисно для великих команд, де на екранах відображаються поточні метрики й звіти, а оператори можуть вчасно помітити

відхилення;

е) *Звіти* (Reports). Хоча основна увага у Zabbix приділяється реальному часу, є можливість створювати автоматизовані звіти:

1) Availability report – відображає середній час доступності (uptime) сервісів за заданий період;

2) Trend report – показує динаміку ключових метрик (CPU, пам’ять, диск) за тривалий проміжок часу;

3) Custom report – адміністратор сам налаштовує з’єднання між метриками та періодами, може генерувати PDF або XLSX-звіти.

Все це дає змогу, з одного боку, контролювати поточний стан у реальному часі, а з іншого – готувати необхідні аналітичні матеріали для керівництва та зовнішніх стейкхолдерів

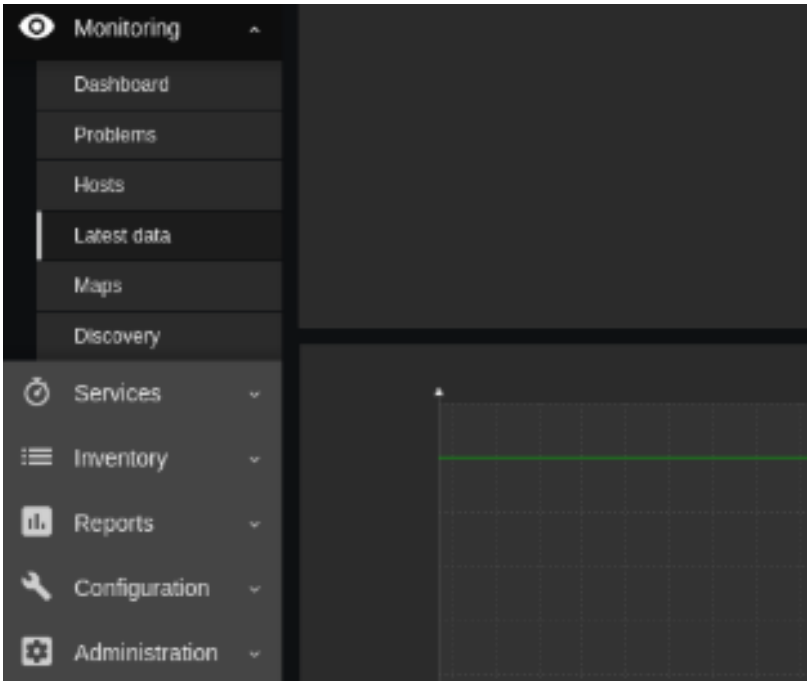


Рисунок 2.2 – Вид на панель керування в Frontend Zabbix

2.6 Безпека та контроль доступу

Встановлення та експлуатація системи моніторингу в корпоративному

середовищі неможлива без належних заходів безпеки. Zabbix передбачає низку механізмів для захисту даних і обмеження доступу:

Шифрування взаємодії (TLS/SSL). Усі компоненти Zabbix (Server, Proxy, Agent, zabbix_sender, zabbix_get) можуть бути налаштовані для обміну даними із застосуванням TLS. Це означає, що передані метрики, а також конфігураційна інформація шифруються, щоб запобігти перехопленню або підміні даних у мережі. Для цього необхідно згенерувати сертифікати (серверний та клієнтський), а далі вказати в конфігураційних файлах параметри:

TLSConnect=psk або TLSConnect=cert.

TLSCAFile, TLSCertFile, TLSKeyFile для вказування шляхів до відповідних ключів.

PSK (pre-shared key) може бути використаний замість сертифікатів у невеликих середовищах для спрощення налаштувань.

Аутентифікація користувачів та розмежування прав (User Roles and Permissions).

Zabbix підтримує ролі та групи користувачів із детальним налаштуванням прав:

― User Roles – дозволяє створювати роль, що дозволяє лише читання даних, роль із можливістю створювати чи редагувати шаблони, роль адміністратора групи хостів тощо.

― User Groups – групи, що містять користувачів із подібними правами (наприклад, група «Operators» може тільки читати дашборди й підтверджувати тригери, але не має прав змінювати конфігурацію; група «Super Admin» має повний доступ).

Можливість дозволити або заборонити доступ до окремих розділів веб-інтерфейсу, наприклад, «Configuration», «Monitoring», «Administration».

Для кожного хоста чи шаблону можна вказати, яка група користувачів матиме до нього доступ (Read, Write, Deny). Це дозволяє ізольовувати дані, наприклад, регіональні адміністратори бачать лише ті хости, за які вони відповідають.

Аудит і журналювання (Auditing and Logging). Zabbix зберігає журнали дій користувачів та системних подій у власній базі даних, що дає змогу відстежувати, хто й коли вносив зміни в конфігурацію, які дії відбулися, які скрипти запускалися, які повідомлення надсилалися. Таким чином можна забезпечити підзвітність і відповідність внутрішнім та зовнішнім аудитам. Залежно від налаштувань, можна зберігати журнали певні дні чи місяці, а старі записи видаляти через механізм Housekeeping

IP-фільтрація та мережеві політики. Часто для серверу Zabbix та проксі встановлюють окремі мережеві сегменти або VLAN. Можна налаштувати, щоб до портів 10050–10051 (порт агента) мали доступ лише певні IP-адреси. Також рекомендується використовувати брандмауери (iptables, firewalld) для обмеження вхідних з'єднань до веб-інтерфейсу (порт 80/443), лише з довірених мереж чи VPN.

Резервне копіювання (Backups). З огляду на важливість конфігурації та накопичених даних, необхідно регулярно робити резервні копії бази даних Zabbix (наприклад, за допомогою mysqldump або pg_dump) та файлів конфігурації /etc/zabbix. Це дає можливість швидко відновити систему після випадкової або навмисної збиткової зміни.

Оновлення та патчі. Для забезпечення безпеки важливо регулярно оновлювати версію Zabbix Server, Proxy, Agent та веб-інтерфейс до останніх стабільних версій. Кожна нова версія містить виправлення критичних вразливостей, а також покращення продуктивності. Оновлення варто проводити у тестовому середовищі, після чого перенести зміни в продакшн.

Завдяки цим механізмам безпеки Zabbix гарантує, що моніторингова інфраструктура є стійкою до зовнішніх загроз та забезпечує суворе розмежування доступу.

2.7 Інтеграція з хмарними сервісами моніторингу

Із сучасним розвитком технологій дедалі більше компаній переходять до

гібридних і мультиклаудних інфраструктур, об’єднуючи власні дата-центри з обчислювальними ресурсами в публічних хмарах. У таких умовах виникає нагальна потреба у єдиній системі моніторингу, яка б давала змогу бачити загальну картину стану всієї інфраструктури – як локальної, так і хмарної. Zabbix як універсальна платформа з відкритим кодом дозволяє реалізувати саме такий підхід, забезпечуючи інтеграцію з основними провайдерами хмарних сервісів.

Ідея інтеграції полягає в тому, що дані з хмарних інстансів, контейнерів чи керованих сервісів (наприклад, обчислювальні потужності EC2 в AWS, служби баз даних RDS, функції Azure Functions або тощо) перетворюються в метрики, які Zabbix може збирати та обробляти, як і будь-які інші дані з агента чи SNMP-пристроїв. Це досягається за допомогою зовнішніх скриптів, плагінів або шаблонів, що використовують REST-API хмарної платформи. Наприклад, офіційні та спільнотні шаблони (`Template AWS CloudWatch`, `Template Azure Monitor`) мають попередньо налаштовані items, які автоматично роблять запити до відповідних API, отримують дані про використання CPU, пам’яті, дискового простору, кількість запитів, час відповіді та інші ключові показники продуктивності, а потім передають їх серверу Zabbix.

Першим етапом інтеграції завжди є налаштування доступу до API хмарного провайдера: отримання необхідних облікових даних (ключі доступу AWS IAM, токени Azure Service Principal, JSON-ключ до облікового запису Google Cloud), налаштування прав доступу, щоб у ході моніторингу не виникало проблем із авторизацією. Далі у Zabbix створюється відповідний користувач або обліковий запис у розділі Administration → Users (наприклад, «AWS_Monitor», «Azure \ Monitor») і встановлюються медіа-типи для сповіщень (email, SMS, Telegram), щоб у разі проблем із хмарними ресурсами можна було одразу отримувати повідомлення.

Після підготовки облікових даних в Zabbix налаштовують «cloud-items» – спеціальні елементи даних, які виконують HTTP-запити до REST-API кожні 60–300 секунд (залежно від інтервалу моніторингу). Ці items, наприклад, aws.ec2.cpuutilization, azure.vm.disk.percentused, gcp.compute.instance.cpu_usage,

отримують числове значення, яке потім зберігається у таблиці history бази даних Zabbix. Для кожного такого елемента можна вказати тип інформації (float або integer), інтервал оновлення, ретеншн-політику (скільки днів зберігати детальні значення, а скільки – агреговані тренди), а також застосувати макроси для автоматичної підстановки імені екземпляра чи регіону.

Наступним кроком є створення тригерів на основі цих даних. Наприклад, вираз {Template AWS CloudWatch:aws.ec2.cpuutilization.last()}>80 стане критичним у разі, якщо середнє завантаження процесора EC2-інстансу вийде за межу 80 %. Інший можливий тригер: {Template Azure Monitor:azure.vm.disk.percentused.last() }>90 – спрацює, якщо заповненість диска віртуальної машини у Azure перевищить 90 %. У Zabbix можна комбінувати декілька умов, наприклад, щоб реагувати лише тоді, коли одночасно завантаженість CPU перевищує 80 % і кількість вільної пам’яті падає нижче певного порогу. Це дає змогу налаштувати складні сценарії моніторингу, що охоплюють не лише ресурсну інформацію, а також дані про стан сервісів.

Для того, щоб не дублювати конфігурацію для кожного окремого екземпляру, доцільно використовувати шаблони. Створюючи окремий шаблон «Template AWS CloudWatch» або «Template Azure Monitor», адміністратор задає універсальні items і тригери, після чого ті застосовуються до декількох хостів, прив’язаних до конкретного облікового запису чи регіону. Шаблон може містити правило макроса, яке підставляє поточний AWS Region або Subscription ID Azure за допомогою зовнішнього скрипта. Після цього до кожного «хмарного» хоста (наприклад, EC2-інстансу) достатньо призначити лише цей шаблон, і Zabbix автоматично почне збирати пульс поточного стану, не потребуючи ручного введення кожного OID чи API-endpoint.

Однією з ключових переваг інтеграції Zabbix з хмарними сервісами є єдиний інтерфейс сповіщень. Незалежно від того, чи відбувається збій у локальному віртуальному середовищі VMware, чи падає продуктивність бази даних у RDS на AWS, всі тригери спрацьовують у Zabbix, формуючи події, які відображаються в єдиному розділі Monitoring → Latest Events. Додатково можна налаштувати групи

користувачів та медіа-типи таким чином, щоб хмарні інциденти призначалися окремим групам DevOps-інженерів або хмарних архітекторів, а локальні – команді системних адміністраторів. Завдяки такому підходу мінімізується ризик пропуску критичних подій у середовищі, що охоплює як on-premise ресурси, так і публічні хмари.

Для забезпечення безперебійного моніторингу потрібно також подумати про безпеку: оскільки REST-API хмарних сервісів передбачає необхідність періодично передавати токени чи ключі доступу, важливо зберігати їх у захищеному сховищі (наприклад, Vault, AWS Secrets Manager), а в самим Zabbix використовувати захищені макроси (Encrypted Macros), які зберігають облікові дані у зашифрованому вигляді. Таким чином, навіть якщо до конфігурації Zabbix матиме доступ стороння особа, вона не зможе прочитати секрети просто з веб-інтерфейсу.

Якщо є необхідність моніторити не лише основні IaaS-сервери, а також PaaS-сервіси (наприклад, AWS Lambda, Azure App Services) або SaaS-сервіси (Office 365, Google Workspace), зазвичай створюють додаткові скрипти, що запитують метрики використання (кількість викликів, час обробки, кількість помилок), розташовують їх у вигляді JSON із чіткою структурою та передають до Zabbix через `zabbix_sender`. У Zabbix створено елемент типу `Zabbix trapper`, який очікує ці дані з CLI. Завдяки `zabbix_sender` можна використовувати складні джерела даних: наприклад, збирати дані про виробничі транзакції в SaaS-сервісах, аналізувати кількість активних користувачів і відображати ці метрики поруч із класичними моніторинговими даними.

Оскільки у багатьох організаціях наразі відсутня чітка межа між on-premise та хмарними ресурсами, інтеграція моніторингу дає змогу створити єдину систему оповіщень та єдиний дашборд, де всі події автоматично корелюються. Наприклад, якщо у локальному дата-центрі падає мережевий комутатор, і одночасно в хмарному середовищі зростає затримка бази даних, адміністратор бачить обидві події, може порівняти час їхнього виникнення та швидко зрозуміти, чи є зв’язок між ними (наприклад, зниження пропускної здатності каналу VPN призвело до затримок у доступі до бази даних).

Завдяки інтеграції Zabbix із хмарними API також спрощується процес масштабування моніторингу: коли до облікового запису AWS додається новий інстанс, зовнішній скрипт, що підтримується функцією auto-registration, може автоматично створити хоста в Zabbix із тегом «cloud» і призначити шаблон «Template AWS CloudWatch», після чого потрібні метрики збиратимуться без додаткового ручного втручання. Це особливо корисно у сценаріях із масштабуванням Auto Scaling Group, де кількість екземплярів може коливатися від кількох до сотень упродовж дня в залежності від навантаження.

Таким чином, інтеграція Zabbix з хмарними сервісами моніторингу не лише дозволяє забезпечити централізоване спостереження за всіма компонентами IT-інфраструктури, але й створює можливість оперативного реагування на відмови й аномалії незалежно від того, де розташовані ресурси – локально чи в публічній хмарі. В результаті такий підхід забезпечує підвищення надійності, скорочення часу простою критичних сервісів і дає змогу гнучкіше керувати розвитком інфраструктури, роблячи процес інтеграції моніторингу невід’ємною частиною стратегії DevOps та хмарної трансформації.

3 ОГЛЯД АЛЬТЕРНАТИВНИХ СИСТЕМ МОНІТОРИНГУ

3.1 Nagios

Nagios Core – це центральний демон (nagios), який працює як планувальник і диспетчер завдань: він періодично опитує плагіни, отримує від них результати і приймає рішення про стан хостів і сервісів на основі конфігураційних файлів (п. а.).

Плагіни у Nagios реалізовані як зовнішні скрипти чи виконувані файли, що повертають код виходу (0 – OK, 1 – WARN, 2 – CRIT, 3 – UNKNOWN), а також рядок із коротким описом проблеми.

Nagios підтримує інтерфейси для agentless моніторингу через протоколи SNMP та WMI, а також клієнт-серверні перевірки через NRPE (Nagios Remote Plugin Executor), NRDP (Nagios Remote Data Processor) або SSH / Telnet (п. а.).

Nagios Core є безкоштовним проектом з відкритим кодом (GPLv2).

Утім Nagios XI – це комерційна надбудова над Core із власним веб-інтерфейсом, базою даних MySQL/MSSQL, розширеним звітуванням, спрощеним процесом налаштування й технічною підтримкою. Nagios XI пропонується за ліцензією з оплатою за вузол (host), що може бути недешево для великих середовищ.

У невеликих середовищах (до 500 хостів) Nagios Core показує стабільну роботу навіть на скромному сервері з 2–4 ГБ RAM.

Для масштабування рекомендується використовувати розподілену архітектуру з розділенням ролей: один основний сервер, декілька резервних серверів, а також декілька екземплярів NRPE/NRDP.

Однак на відміну від Zabbix, у Nagios немає вбудованої підтримки власних проксі-серверів для розвантаження каналу чи кешування метрик. Масштабування часто потребує ручного налаштування кластерів (High Availability) та додаткових інструментів, як-от Director (для Icinga).

Налаштування Nagios здійснюється через плоскі конфігураційні файли

(hosts.cfg, services.cfg), де адміністратор вручну описує всі хости та сервіси.

Для автоматизації часто використовують такі зовнішні інструменти, як Puppet, Chef, Ansible або Director (Icinga Director для форку Icinga2).

Nagios не підтримує «Low-Level Discovery (LLD)» «з коробки», тому в середовищах із динамічними або великомасштабними змінними ресурсами (контейнери, віртуальні машини) адміністратору доводиться додавати вручну кожен елемент або використовувати зовнішні скрипти для генерації конфігураційних записів.

Nagios Core забезпечує базовий веб-інтерфейс із переглядом статусу хостів, сервісів, історією подій і простою табличною формою даних.

Nagios XI додає розширений GUI з готовими шаблонами звітів, аналітикою SLA, автоматичним плануванням звітів і можливістю експорту в PDF/CSV.

На відміну від Zabbix, який має вбудовані потужні дашборди, карти мережі, інтерактивні графіки та можливість створювати настроювані віджети без додаткових модулів, у Nagios потрібно встановлювати сторонні плагіни, як-от NagVis для побудови мережевих карт або PNP4Nagios для розширених графіків.

Nagios підтримує базову аутентифікацію через веб-сервер (Apache HTTP Basic Auth) і обмеження доступу до окремих сторінок шляхом налаштування файлу .htaccess.

У Nagios XI додано розширені механізми ролей і прав доступу, інтеграцію з LDAP/AD, шифрування з’єднань із базою та маршрутизацію повідомлень через TLS.

3.2. Icinga

Icinga виник як форк Nagios у 2009 році, метою якого було покращення продуктивності та створення розширеного веб-інтерфейсу на базі модулів Icinga Web.

Основна архітектура Icinga 2 передбачає кілька компонентів: icinga2 daemon (моніторинг ядра), IDO (Icinga Data Output) для запису даних у базу, Icinga Web 2 для GUI і Icinga Director для управління конфігураціями через API.

Icinga 2 є безкоштовним проектом з відкритим кодом (GPLv2), з підтримкою спільноти.

Комерційних ліцензій немає, але існують платні підписки на Enterprise-пакети підтримки від Icinga GmbH, які забезпечують доступ до професійної технічної допомоги.

Icinga 2 спроектований із урахуванням кластеризації: можна створювати distributed monitoring-кластер із декількох екземплярів icinga2, які взаємодіють через API, забезпечуючи високу доступність (HA) та розподіл навантаження.

У порівнянні з Nagios, Icinga 2 підтримує асинхронне виконання перевірок (завдяки використанню бібліотеки Twisted Reactor), що пришвидшує обробку великої кількості діагностичних запитів.

Проте для горизонтального масштабування в Icinga 2 доводиться налаштовувати окремі зони (zones) та endpoints, що потребує певних зусиль. У Zabbix же є вбудовані Proxy, які «по суті» є інстансами збирачів, та кілька серверів моніторингу, які можуть працювати в харденому кластері.

Icinga Director (плагін для Icinga Web 2) дає змогу автоматизувати створення хостів, призначення шаблонів, налаштування груп і властивостей через інтерактивний UI або REST API.

На додаток до Director часто використовують Puppet, Chef, Ansible для управління конфігураціями, у яких описуються всі хости й сервіси як code.

У Zabbix автоматизація відбувається через LLD, авто-реєстрацію агента, REST API, а також через Terraform-провайдер (Community) для опису всього стеку Zabbix як коду.

Icinga Web 2 надає базовий набір дашбордів, перегляд статусу хостів, сервісів, журнал подій.

Для більш просунутої візуалізації рекомендують інтегрувати Icinga з Grafana або Graphite, які підключаються до Icinga IDO (бази даних) через проміжні

конектори.

Icinga Web 2 підтримує LDAP/AD, Role-Based Access Control (RBAC) із налаштуванням рівнів доступу до хостів, сервісів та конфігурацій через ролі (п. а.).

Шифрування взаємодії між агентом та сервером (TLS) налаштовується через директорію /etc/icinga2/certs.

У Zabbix механізм безпеки охоплює TLS-шифрування між усіма компонентами, детальніше розмежування прав для об’єктів (read/write/deny на рівні хоста, шаблону, карти), а також аудит дій користувачів (логування операцій у базі даних).

Масштабування: Icinga 2 підтримує кластеризацію через zones/endpoints, але потребує ручного налаштування; Zabbix має вбудовані Proxy та можливості HA для серверів основного моніторингу

Автоматизація: Icinga потребує Icinga Director чи сторонніх IaC-інструментів для автоматизації; Zabbix забезпечує LLD, API та авто-реєстрацію агента .

Візуалізація: Icinga базується на інтеграції з Grafana/Graphite; Zabbix пропонує «все-в-одному» frontend із вбудованими дашбордами

Безпека: обидві платформи підтримують TLS і RBAC; Zabbix додає докладний аудит операцій і granular права доступу до об’єктів

3.3. Prometheus

Prometheus – time-series база даних із відкритим кодом, спочатку розроблена в SoundCloud у 2012 р. Першопричиною появи було забезпечити моніторинг динамічних мікросервісних середовищ (Kubernetes, Docker).

Його архітектура базується на кількох компонентах:

Prometheus Server – збирає метрики через HTTP pull (scraping) за вказаними в конфігурації targets (експортери або сервис discovery).

Exporters – проміжні процеси, які перетворюють внутрішні показники систем (Linux, SNMP, JMX, PostgreSQL, Docker, Kubernetes) у формат, зрозумілий Prometheus (exposition format).

Alertmanager – обробляє алерти, вирішує маршрутизацію, групування та кесонування (silencing) повідомлень, інтегрується з email, PagerDuty, Slack тощо.

Pushgateway – компонент, що дає змогу «push» певним сервісам (batch-job) одноразово надіслати метрики, коли pull-модель неможлива.

Client Libraries – бібліотеки для Golang, Java, Python, Ruby тощо, щоб інструмент інтегрувався безпосередньо в додаток і виставляв метрики.

Prometheus є повністю безкоштовною системою з відкритим кодом (Apache 2.0).

Для підприємств існують керовані хмарні сервіси (Prometheus as a Service) від компаній, як Lightstep, MetricFire, Grafana Labs, але сам Prometheus не вимагає витрат на ліцензію.

Prometheus орієнтований на збір метрик із високою частотою (до 1 запиту в секунду) та оптимізований для обробки десятків тисяч targets.

Однак його вбудоване сховище обмежене: зазвичай один екземпляр Prometheus працює з даними тільки за останні 15 днів (за умовчанням). Для довготермінового зберігання використовують рішення як Cortex, Thanos, VictoriaMetrics або зберігають проміжні тренди в InfluxDB/TimescaleDB.

Для масштабування Prometheus допускає горизонтальні кластери з federation, де один «головний» збирає агрегації з декількох «регіональних» екземплярів.

У порівнянні з Zabbix, який має єдину централізовану базу SQL із розбиттям на history / trends, Prometheus потребує додаткових систем для довготермінового зберігання та складної агрегації.

Prometheus відмінно працює в Kubernetes: у нього є вбудований service discovery, що дозволяє автоматично знаходити нові Pod, Service, Endpoints, Ingress.

Exporters для Kubernetes (kube-state-metrics, node-exporter) дозволяють збирати детальні метрики на рівні кластера й вузлів.

У Zabbix автоматичне виявлення обмежується LLD на базі регулярних виразів і SNMP-discovery, але для Kubernetes потрібні зовнішні інтеграції (Zabbix Operator).

Prometheus постачається з власним UI, яке дозволяє робити ad hoc запити

PromQL і малювати базові графіки.

Найчастіше для візуалізації Prometheus використовують Grafana, де можна створювати кастомні дашборди, комбінувати дані з різних джерел і застосовувати складні трансформації.

Alerter Prometheus – це Alertmanager, який має можливість групувати сповіщення, робити derouting та доставляти повідомлення через Slack, email, webhook. У Zabbix аналогічну роль виконують Actions з макросами та escalation.

Prometheus UI зазвичай відкритий лише всередині приватної мережі; підтримка безпеки через TLS та базову HTTP аутентифікацію налаштовується за допомогою проксі (NGINX, OAuth2 Proxy).

Alertmanager підтримує аутентифікацію через веб-конфігурацію і може інтегруватися з LDAP через middleware.

У Zabbix безпека вбудована: зв’язок між агентами, проксі та сервером підтримує TLS, фронтенд має RBAC і інтеграцію з LDAP/AD.

3.4. PRTG Network Monitor

PRTG Network Monitor (Paessler) – це комерційна платформа, побудована як єдиний інсталятор («all-in-one») для Windows Server.

Основна одиниця моніторингу – «сенсор» (sensor): кожен сенсор – це окремий параметр (наприклад, CPU, пам’ять, HTTP-запит, SNMP OID, NetFlow, sFlow, WMI, REST-API тощо).

Архітектурно PRTG складається з «core-сервера», «probe-серверів» (локальних або віддалених), веб-інтерфейсу IIS-основи і бази даних (MS SQL).

PRTG пропонує безкоштовну версію до 100 сенсорів.

Далі ліцензування йде по моделлю «кількість сенсорів»: 500, 1000, 2500 сенсорів, тощо, де базова вартість за 500 сенсорів може починатися від кількох сотень доларів на рік.

PRTG легко масштабується за рахунок розподілення «probe-серверів»: для

кожного «probe» налаштовують окремий сервер (на Windows або Linux через Mono), який збирає дані локально і передає їх центральному серверу.

Централізована база даних MS SQL може призводити до швидкого зростання розміру при великій кількості сенсорів, тому рекомендують застосовувати ретеншн-політику та архівування даних.

PRTG підтримує автоматичне виявлення пристроїв у мережі через SNMP, WMI, UPnP та HTTP протоколи.

Автоматичне виявлення дозволяє швидко створити інвентаризацію пристроїв і автоматично призначити сенсори за шаблоном (наприклад, для кожного Windows Server – WMI-сенсори, для кожного маршрутизатора – SNMP-сенсори).

У Zabbix аналогом є SNMP Discovery і LLD, але Zabbix не має вбудованого «probe-серверу» для Windows – натомість використовує стандартизований Proxy або агент.

PRTG має зручний веб-інтерфейс із готовими дашбордами, картами мережі (Network Maps) та докладними звітами (uptime, тривалість збою, продуктивність сенсорів).

У візуалізації PRTG часто використовують готові шаблони карт для rack view (стійки), floor plan (кімнати) і географічні карти, куди можна поміщати пристрої з індикаторами стану.

Zabbix пропонує подібні карти та панелі, але PRTG вирізняється своєю інтерактивністю та простотою налаштування мережевих карт без необхідності сторонніх модулів.

PRTG підтримує декілька рівнів користувацьких прав: читання, читання+зміна, адмін, з можливістю налаштування доступу до окремих груп пристроїв або сенсорів.

Для захисту використовується HTTPS, LDAP/AD-інтеграція, multi-factor authentication, а також шифрування з’єднань між probe-сервером і core-сервером.

Zabbix підтримує TLS-шифрування всіх каналів між агентами/проксі і сервером, LDAP/AD, RBAC та детальний аудит користувачів.

3.5. SolarWinds Orion

SolarWinds Orion – комплексна платформа, що складається з набору модулів: NPM (Network Performance Monitor), SAM (Server & Application Monitor), NCM (Network Configuration Manager), VNQM (Voice & Network Quality Manager), VMAN (Virtualization Manager) тощо.

Основною базою даних є SQL Server, куди записуються всі метрики та події. Орієнтована на Windows Server (або SQL Server на Linux), платформа використовує службу ICMP, SNMP, WMI, Flow-протоколи, як NetFlow / IPFIX, для збору метрик.

SolarWinds Orion – комерційне рішення з оплатою за кількість елементів (Nodes, Interfaces, Volumes) і придбанням окремих модулів.

Вартість може сягати десятків тисяч доларів на рік залежно від розміру інфраструктури та кількості ліцензій.

Для великих мереж Orion дозволяє розгортати додаткові Polling Engines, щоб розвантажити основний сервер.

Перевагою є централізація всіх модулів в одному UI, але через масштабування SQL Server можуть виникати затримки при обробці великих обсягів даних.

Zabbix масштабується через Proxy/Server HA, без необхідності SQL Server Enterprise-рівня; використання open-source СУБД дає гнучкість у виборі архітектури.

Orion підтримує SNMP Walk, NMAP Discovery, WMI Discovery, подібно до PRTG; адміністратор може настроїти авто-виявлення за IP-діапазоном.

Zabbix використовує SNMP Discovery і LLD, але потребує налаштування правил (regex) та реакцій, що потребує дещо більше ручної роботи.

Orion NPM забезпечує інтерактивні карти мережі з drag-and-drop, «NetPath» (визначає маршрути до хоста та вимірює затримки), «PerfStack» (кореляція метрик у Timeline).

Zabbix має Network Maps, проте функціонал «NetPath» або «PerfStack» відсутній «з коробки». Для подібних можливостей у Zabbix потрібно інтегрувати

Grafana.

Orion підтримує LDAP/AD, ролі й групи в межах SIEM-інтеграцій, ефективно працює з TLS, а також надає audit trails.

Zabbix пропонує TLS-шифрування між агентами, проксі і сервером, LDAP/AD та детальне RBAC, що дозволяє створювати обмежені групи з granular правами доступу.

3.6. Datadog

Datadog – це хмарна SaaS-платформа для моніторингу, логування й APM (Application Performance Monitoring), яка використовує агента Datadog Agent, бібліотеки (DogStatsD) для збору метрик і Logs Forwarder для логів (п. а.).

Datadog Agent легко встановлюється на Linux, Windows, macOS, контейнери та збирає метрики системи, логи й події, а також траси HTTP (APM).

Datadog працює за підпискою – вартість залежить від кількості хостів, кількості контейнерів, обсягу логів і потрібних модулів (APM, Security, Synthetic).

Загальні витрати можуть бути значно вищими, ніж у самохостингового Zabbix, особливо при великому обсязі даних і складній інфраструктурі.

Оскільки Datadog – це SaaS, масштабування зазвичай відбувається автоматично з боку провайдера, і клієнту не потрібно турбуватися про продуктивність баз даних.

Він надає near-real-time моніторинг із частотою ~15 с для Host-agent checkpoints і до 1 с для деяких контейнерних метрик.

Zabbix у цій сфері може збирати метрики з інтервалом до 1 с у Active Check, але для зберігання таких даних потрібне добре налаштоване серверне середовище.

Datadog Agent підтримує понад 450 готових інтеграцій (AWS, Azure, GCP, Kubernetes, Docker, PostgreSQL, MySQL, Redis, Kafka тощо), що робить його чудовим вибором у середовищах “хмара” та мікросервісів.

Zabbix також підтримує багато інтеграцій через плагіни, шаблони і зовнішні скрипти, але часто потребує більше часу на налаштування та підтримку менш “однорідного” інфраструктурного стеку.

Datadog надає в реальному часі дашборди з drag-and-drop побудовою графіків, heatmap, anomaly detection, service maps (граф об’єктів і їхніх залежностей), APM діаграми запитів і flame graphs.

Zabbix може створювати схожі графіки та карти, але для APM та trace-аналізу йому не вистачає вбудованих можливостей: необхідно залучати зовнішні інструменти (Jaeger, Zipkin) і інтегрувати їх із Zabbix через скрипти.

Datadog дотримується стандартів SOC 2, HIPAA, підтримує SAML, багато факторну аутентифікацію, шифрування TLS/SSL, ізольовані VPC.

Zabbix гарантує безпеку через TLS, LDAP/AD і ролі із granular правами; але оскільки платформа self-hosted, зобов’язання щодо відповідності нормативам лежать на організації користувача.

3.7. Grafana

Grafana – це open-source платформа для створення дашбордів і візуалізації метрик з мульті-джерел (Prometheus, InfluxDB, Elasticsearch, MySQL, PostgreSQL, Zabbix, Graphite тощо).

Вона складається з бекенду (Go), фронтенду (TypeScript/React), а також плагінів (DataSource Plugins, Panel Plugins), що дозволяє підключати нові джерела і типи віджетів.

Grafana пропонується у двох основних варіантах: Grafana OSS (Apache 2.0) – безкоштовна версія, яку можна встановити «on-premise», та Grafana Enterprise (комерційна), що надає додаткові плагіни, ситеми SSO, аудит дій і преміальну підтримку від Grafana Labs.

Grafana використовується як фронтенд, тому продуктивність головним чином залежить від обраного сховища метрик: для Prometheus / InfluxDB масштабування виконується на рівні цих баз; для Zabbix необхідний плагін Zabbix-DataSource, який

робить запити до Zabbix API.

Для великих дашбордів рекомендується використовувати Grafana Enterprise або кластер Prometheus / Thanos, щоб зменшити навантаження на окремі інстанси.

Плагін Zabbix-DataSource дозволяє Grafana безпосередньо запитувати інтерфейс Zabbix API для отримання даних про елементи (items), об’єднання із тригерами та побудови графіків.

Це дає змогу будувати складні дашборди з метриками Zabbix у Grafana, використовуючи усі переваги Grafana (templating, transformations, alerts, annotations).

ВИСНОВКИ

У бакалаврській роботі проведено комплексне дослідження теоретичних та практичних аспектів побудови моніторингових систем у корпоративних мережах. Докладніше обґрунтовано та розглянуто використання МС Zabbix. Основними результатами роботи є:

– проведено детальний аналіз ролі моніторингу в корпоративних мережах, обґрунтовано потребу у безперервному стеженні за станом IT-інфраструктури;

– розглянуто класифікацію типів моніторингу (мережевий, серверний, APM, безпека), а також методи збору даних (агентський, агентless, trap-орієнтований, веб-моніторинг, лог-моніторинг);

– описано архітектуру Zabbix як прикладу сучасної системи моніторингу: основні компоненти (Server, Proxy, Agent, Frontend, база даних), механізми обробки даних (items, triggers, actions), а також засоби автоматизації конфігурації (шаблони, LLD, auto-registration агента, API);

– реалізовано практичний прототип системи на базі Docker Compose: розгорнуто сервер Zabbix з MySQL-базою, конфігуровано веб-інтерфейс, під’єднано агенти та SNMP-пристрої, автоматично додано хости через шаблони й LLD;

– налаштовано правила сповіщень і тригери для типових сценаріїв (перевантаження CPU, брак дискового простору, недоступність MySQL, уповільнення роботи веб-сервера), розроблено дії з escalation-логікою (email, SMS, remote command);

– створено кастомізований дашборд для оперативного моніторингу критичних показників, включаючи карти мережі, графіки CPU/пам’яті, табличний огляд останніх даних і фільтр «Problem only» для тригерів;

– проведено тестування на прикладі кейсів із навмисним створенням проблем: заповнення диска, зупинка MySQL-служби, зупинка веб-сервера; у кожному випадку система миттєво виявляла аномалію, формувала подію й надсилала сповіщення, підтверджуючи працездатність механізму моніторингу;

– виконано порівняльний аналіз альтернатив (Nagios, Icinga, Prometheus, PRTG, SolarWinds, Datadog, Grafana), у результаті якого встановлено, що Zabbix є універсальним open-source-рішенням з гнучкою автоматизацією, вбудованими дашбордами й достатнім набором шаблонів, що спрощує розгортання в класичних on-premise середовищах.

Отже, проведене дослідження підтвердило, що Zabbix здатен стати ефективним інструментом моніторингу корпоративних мереж: поєднуючи в собі легкість початкового запуску (особливо за допомогою контейнеризації), достатній рівень автоматизації (LLD, auto-registration, REST API), широкі можливості візуалізації (дашборди, карти мережі, графіки) та гнучку систему сповіщень (actions із escalation). Найбільшою перевагою Zabbix є можливість централізовано контролювати різнорідні компоненти інфраструктури (Linux-сервери, Windows-машини, SNMP-пристрої, веб-сервіси) без значних ліцензійних витрат, адже платформа лишається безкоштовною і відкритою.

У перспективі рекомендується звернути увагу на такі напрями вдосконалення:

— інтеграція Zabbix із зовнішніми системами аналітики (Grafana, Elasticsearch) для поглибленого аналізу трендів і побудови більш складних звітів;

— використання алгоритмів машинного навчання та AIOps-підходів для автоматичного виявлення аномалій і прогнозування потенційних відмов;

— розширення моніторингу на мультиклаудові середовища, Kubernetes-кластери та безсерверні сервіси, де необхідна динамічна конфігурація джерел метрик;

— впровадження механізмів управління конфігурацією як код (Terraform, Ansible), що забезпечить повну відтворюваність і контроль версій налаштувань Zabbix;

Таким чином, рішення демонструє доцільність використання Zabbix у умовах класичної корпоративної мережі: дозволяє забезпечити доступність сервісів, зменшити час реагування та створити основу для подальшого розвитку моніторингу з урахуванням майбутніх потреб інформаційної інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About - icinga 2. *Open Source Monitoring for Complex IT Environments | Icinga*. URL: https://icinga.com/docs/icinga-2/latest/doc/01-about/

2. Cloud monitoring as a service | datadog. *Datadog*. URL: https://www.datadoghq.com/.

3. Documentation for SolarWinds Platform. *Documentation for SolarWinds*. URL: https://documentation.solarwinds.com/en/success_center/orionplatform/content/orion_platform_documentation.htm

4. Documentation. *Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution*. URL: https://www.zabbix.com/manuals

5. Getting started with PRTG - Academy modules. *Paessler - The Monitoring Experts*. URL: https://www.paessler.com/support/getting-started

6. Grafana: The open and composable observability platform | Grafana Labs. *Grafana Labs*. URL: https://grafana.com/

7. Nagios: free open source IT monitoring tools. *Nagios Open Source*. URL: https://www.nagios.org/

8. Overview | prometheus. *Prometheus - Monitoring system & time series database*. URL: https://prometheus.io/docs/introduction/overview/

9. What are the principles of monitoring and evaluation?. *EvalCommunity*. URL: https://www.evalcommunity.com/career-center/principles-of-monitoring-and-evaluation/

10. Рогнєда Княжина. Опановуємо Docker: Від ізольованих контейнерів до масштабного кластера на AWS. robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс. URL: https://robotdreams.cc/uk/blog/607-docker-basics-part-1.

11. Docker: accelerated container application development. Docker. URL: https://www.docker.com.

12. NetworkChuck. Monitor all your stuff RIGHT NOW!!, 2024. YouTube. URL: https://www.youtube.com/watch?v=-2yzXSIuC8o.

13. NetworkChuck. The hacker’s roadmap (how to get started in IT in 2025), 2025. YouTube. URL: https://www.youtube.com/watch?v=5xWnmUEi1Qw.

14. NetworkChuck. You need to learn Docker RIGHT NOW!! // Docker Containers 101, 2020. YouTube. URL: https://www.youtube.com/watch?v=eGz9DS-aIeY.

15. Network monitoring 101: how to monitor networks effectively | splunk. Splunk. URL: https://www.splunk.com/en_us/blog/learn/network-monitoring.html.

16. Ratan V., Li K. F. NetFlow: network monitoring and intelligence gathering. SpringerLink. URL: https://link.springer.com/chapter/10.1007/978-3-319-49109-7_83.

17. Schirrmacher D. Security updates: Network monitoring tool Zabbix offers attack surface. Security. URL: https://www.heise.de/en/news/Security-updates-Network-monitoring-tool-Zabbix-offers-attack-surface-10337540.html.

18. Visualizing network performance of your AWS cloud workloads with network flow monitor | amazon web services. Amazon Web Services. URL: https://aws.amazon.com/ru/blogs/networking-and-content-delivery/visualizing-network-performance-of-your-aws-cloud-workloads-with-network-flow-monitor/.

19. What is a container?. Docker Documentation. URL: https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container/.

20. What is docker?. fw_error_www. URL: https://www.oracle.com/ua/cloud/cloud-native/container-registry/what-is-docker/.

21. 21 best network monitoring tools for 2025 (free & paid). Comparitech. URL: https://www.comparitech.com/net-admin/network-monitoring-tools/.

22. Dkk K. All-in-One Zabbix Blog: The Ultimate Guide to Monitoring Everything. Medium. URL: https://medium.com/devsecops-community/all-in-one-zabbix-blog-the-ultimate-guide-to-monitoring-everything-c31ad804af9c.

23. Dkk K. All-in-One Zabbix Blog: The Ultimate Guide to Monitoring Everything. Medium. URL: https://medium.com/devsecops-community/all-in-one-

zabbix-blog-the-ultimate-guide-to-monitoring-everything-c31ad804af9c.

24. Getting Started with Zabbix – Hosts, Items, and Triggers - Zabbix Blog. Zabbix Blog. URL: https://blog.zabbix.com/getting-started-with-zabbix-hosts-items-and-triggers/30190/.

25. Maxfield F. Simplified zabbix install: step-by-step with docker and portainer. Medium. URL: https://medium.com/@fredrik.maxfield/simplified-zabbix-deployment-step-by-step-with-docker-and-portainer-19e85c08a65b.

26. OKLABS. Lecture 89 - Install Zabbix Server using docker compose, 2024. YouTube. URL: https://www.youtube.com/watch?v=6p7B6qBtnos.

27. Zabbix – Система моніторингу серверів та додатків | Zomro. Zomro.com | Reliable Hosting | VPS/VDS server at an affordable price. URL: https://zomro.com/ua/marketplace/zabbix.

28. Zabbix ⇒ 25 років на ринку IT послуг для бізнесу - TechExpert. TechExpert. URL: https://techexpert.ua/it-products/zabbix/.

29. Zabbix Monitoring: A Full Guide to Setting Up and Using Zabbix - . DediRock - Global Dedicated & KVM VPS - Everything You Need Nothing You Dont -. URL: https://dedirock.com/blog/zabbix-monitoring-a-full-guide-to-setting-up-and-using-zabbix/.

30. Zabbix server 7 in docker compose - DevOps - Oleksandr Liutyi's WIKI. Dashboard - Oleksandr Liutyi's WIKI. URL: https://wiki.liutyi.info/display/DEVOPS/Zabbix+server+7+in+docker+compose.

Додаток

ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

(презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ



Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ
БАКАЛАВРА

на тему: «Моніторингові системи у корпоративних мережах»

Виконав студент групи КІД-41
Гітченко Євгеній Олегович

Керівник кваліфікаційної роботи:
Борова Юлія Олександрівна, асистент кафедри

Київ – 2025

Мета роботи – Розгортання та налаштування моніторингової системи Zabbix для корпоративної комп'ютерної мережі

Об`єкт дослідження – Корпоративна комп'ютерна мережа з централізованим моніторингом її компонентів

Предмет дослідження – Аналіз системи моніторингу як засобу забезпечення стабільності та керованості IT-інфраструктури на прикладі системи Zabbix

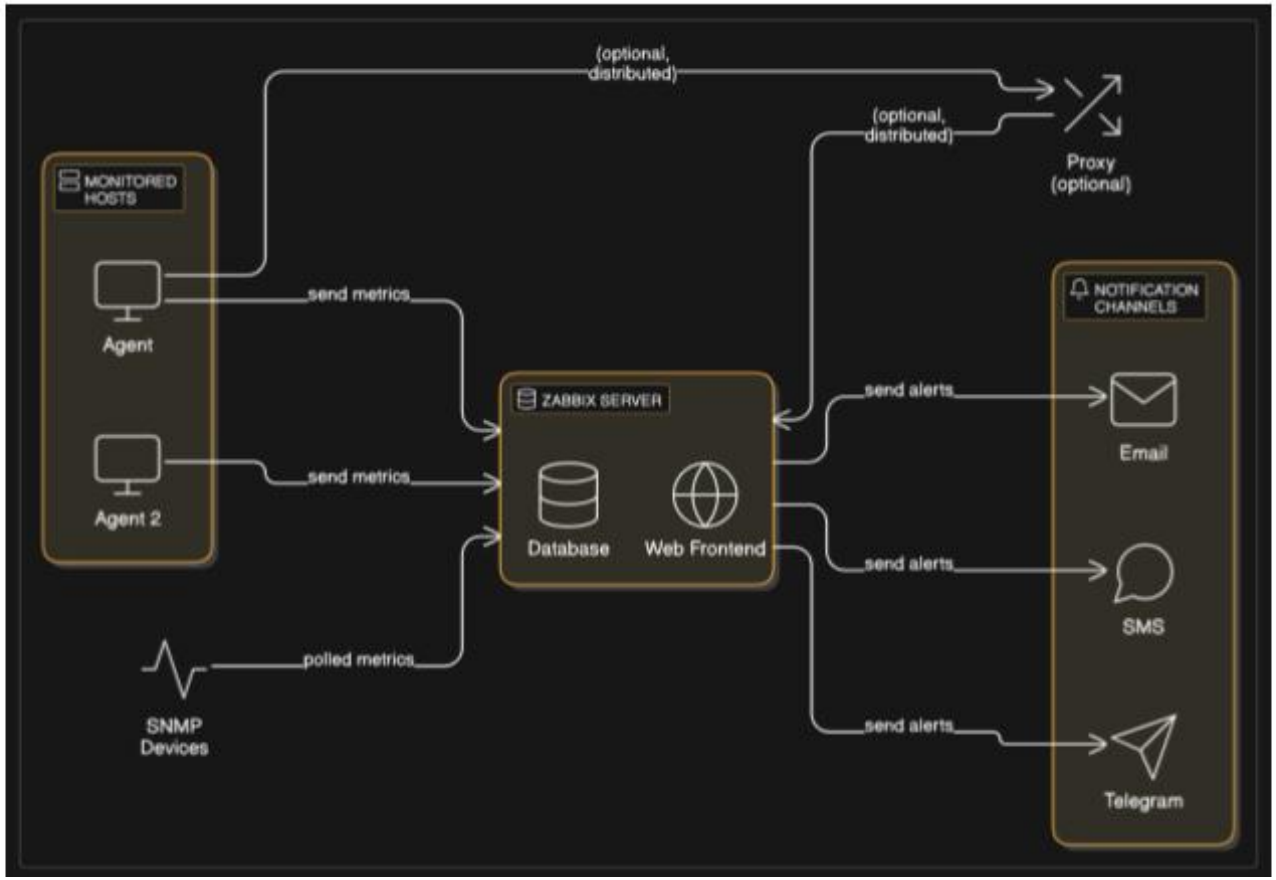
Актуальність обраної теми

У сучасних умовах збої в роботі сервісів можуть призводити до фінансових втрат, зупинки бізнес-процесів або втрати репутації.

Моніторинг дозволяє не тільки виявляти проблеми до того, як вони стануть критичними, але й автоматично реагувати — надсилати сповіщення або навіть перезапускати сервіси.

Саме тому впровадження якісної системи моніторингу є обов’язковим елементом сучасної корпоративної мережі.

Архітектура Zabbix



Агенти



OS DISTRIBUTION	OS VERSION	HARDWARE	ZABBIX VERSION	ENCRYPTION	PACKAGING
Windows	Any	amd64	7.2	OpenSSL	MSI
Linux		i386	7.0 LTS	No encryption	Archive
macOS			6.2		
AIX			6.0 LTS		
FreeBSD			5.4		
OpenBSD			5.2		
Solaris			5.0 LTS		
			4.4		
			4.2		
			4.0 LTS		
			3.0 LTS		

zabbix_agent-7.2.6-windows-amd64-openssl.msi
8,9 МБ • Готово

Система сповіщень і проксі

Zabbix проксі - це процес, який може збирати дані моніторингу з одного або декількох пристроїв і надсилати інформацію на сервер Zabbix, по суті, працюючи від імені сервера. Всі зібрані дані буферизуються локально, а потім передаються на сервер Zabbix, до якого належить проксі.

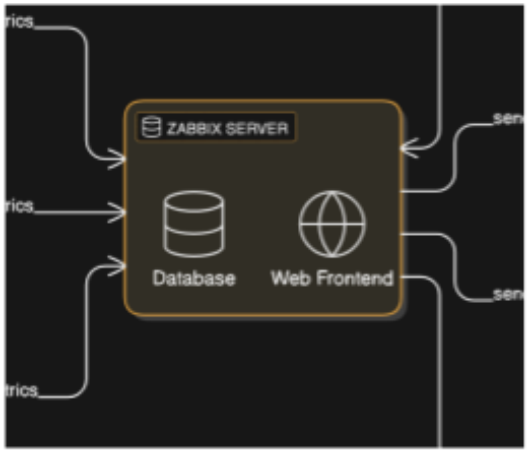
Media types

Name ▲	Type	Status	Used in actions	Details
Email	Email	Enabled		SMTP server: "mail.zabbix.com",
Mattermost	Webhook	Enabled		
Opsgenie	Webhook	Enabled		



Zabbix сервер, база даних і фронтенд





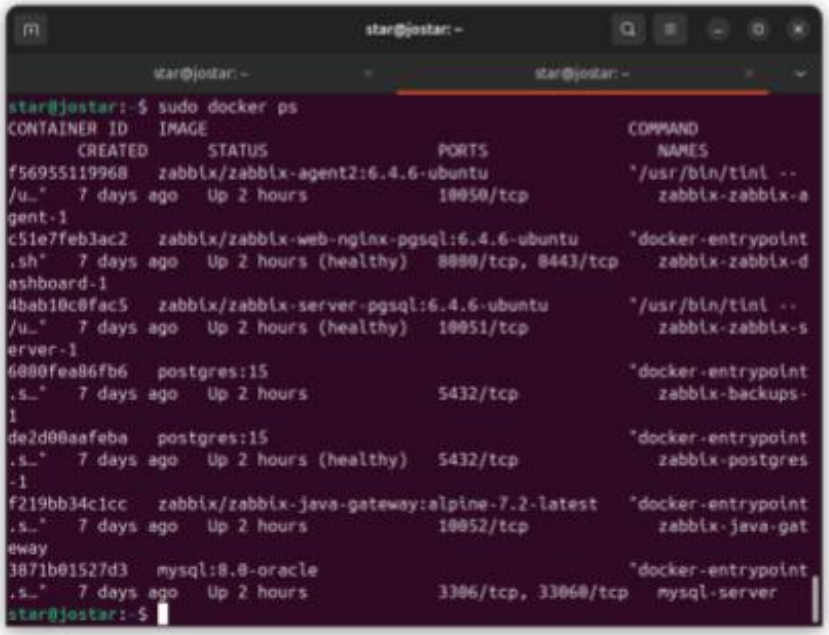


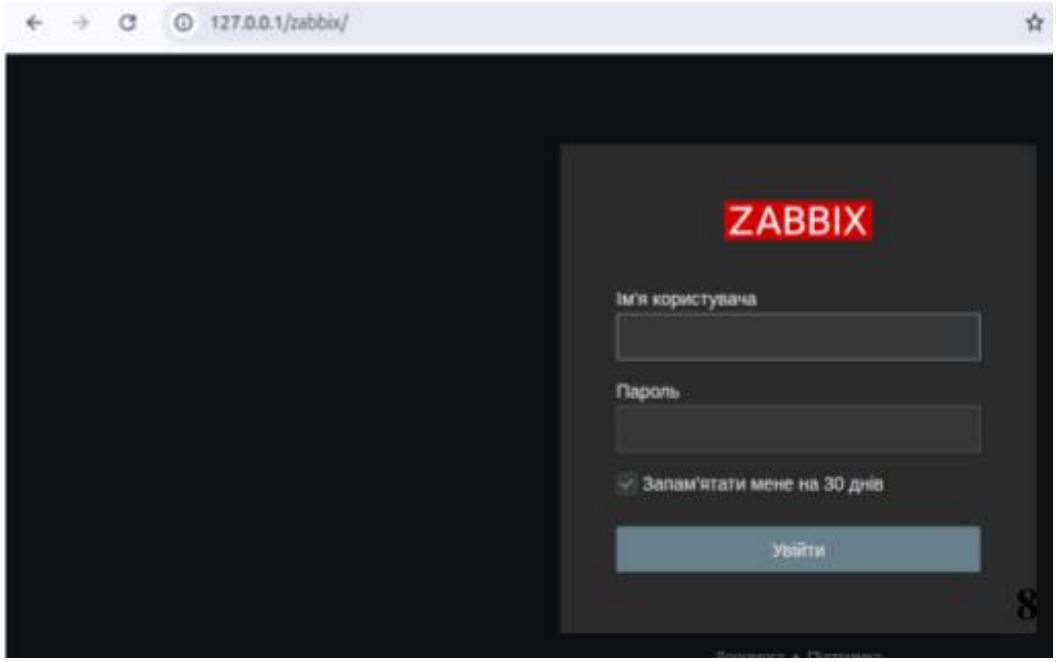




Docker

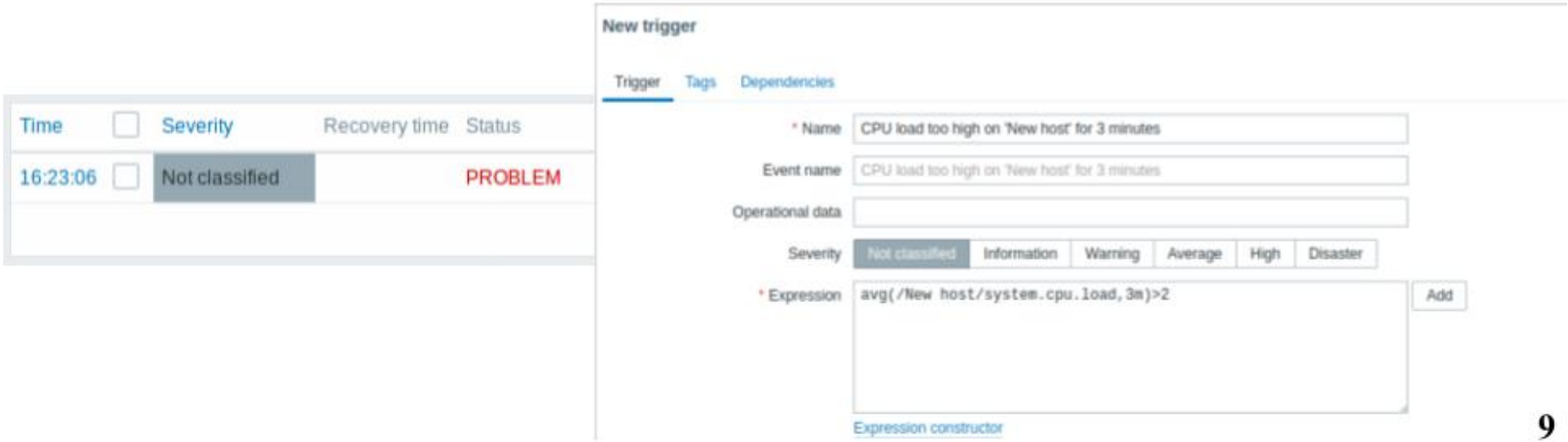
Ця робота була виконана безпосередньо на машині, так і на Docker.
Для розгортання я використав docker-compose. Це дозволило мені швидко створити повноцінне середовище: сервер Zabbix, MySQL-базу, веб-інтерфейс





Тригери

Для перевірки я імітував проблеми: перевантаження процесора, зникнення мережі, переповнення диска. У всіх випадках система правильно відпрацьовувала тригери, надсилала повідомлення, а при потребі запускала автоматичні дії.



Висновки

У підсумку, в кваліфікаційній роботі було реалізовано дієву систему моніторингу на базі Zabbix, що дозволяє контролювати критичні вузли мережі, прогнозувати потенційні збої й автоматизувати реакцію.

Система показала себе як надійна, гнучка і достатньо потужна навіть для розширеного середовища.

Таке рішення може бути легко адаптоване до умов реального підприємства — без великих витрат і з хорошими результатами вже на старті.