Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра__
(бакалавра, спеціаліста, магістра)

студента Денисенка Олексія Дмитровича
(ПІБ)
академічної групи 123м-20-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему «Програмно-технічна реалізація засобів комп'ютерної системи контролю завантаженості веб-серверу ІТ-компанії Civenty»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Каштан В.Ю.			
розділів:				
синтез системи	доц. Ткаченко С.М.			
розроблення програмного забезпечення	ас. Бешта Л.В.			
Рецензент	доц. Сафаров О.О.			
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище,ініціали)

«_» _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістр

студенту Денисенко О.Д. **академічної групи** 123м-20-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему «Програмно-технічна реалізація засобів комп'ютерної системи контролю завантаженості веб-серверу IT-компанії Civenty»

затверджену наказом ректора НТУ «Дніпровська політехніка» від 10.12.2021 р. №1036

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	20.09.2021
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	25.10.2021
Синтез системи	Розробка комп'ютерної системи	15.11.2021
Розроблення програмного забезпечення	Розробка програмного забезпечення	29.11.2021
Експериментальний розділ	Проведення і обробка результатів експериментів	15.12.2021
Графічна частина	Графічні результати роботи подати у вигляді рисунків, схем, таблиць на 20 арк. Формату А4.	05.01.2022

Завдання видано _____ доц. Каштан В.Ю.
(підпис керівника) (прізвище, ініціали)
Дата видачі 06 вересня 2021р.
Дата подання до екзаменаційної комісії 10.01.2022 р.
Прийнято до виконання _____ Денисенко О.Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 83 с., 22 рис., 7 табл., 23 джерела, 1 додаток.

Об’єкт розробки: серверна кімната головного офісу IT-компанії Civenty.

Мета роботи: розробити комп’ютерну систему контролю завантаженості веб-серверу IT-компанії Civenty. Обґрунтувати використання власної системи моніторингу поміж існуючих рішень.

Пояснювальна записка має аналіз існуючих систем моніторингу, описуючи їх основні недоліки та переваги. За допомогою цих даних було сформульовано завдання дослідження. Після цього, необхідно вирішити було наукове завдання побудувавши модель системи контролю завантаженості веб-серверу. Наступним кроком потрібно було формулювання технічних вимог до створюваної системи, за допомогою яких було побудовано результуючу структурну схему системи моніторингу. Далі була проведена розробка програмного забезпечення на основі побудованих схем алгоритмів. До цього ж, описуючи зв’язок між програмами та їх функціональних можливостей. Останнім кроком було проведення експерименту створеної системи контролю завантаженості веб-серверу.

МОНІТОРИНГ, ВЕБ-СЕРВЕР, DOCKER, DOCKER-COMPOSE, КОНТЕЙНЕР, ANSIBLE

ABSTRACT

Explanatory note: 83 pages, 22 figures, 7 tables, 23 sources, 1 application.

Object of research: A head office’s server room of an IT-company Civenty.

Objective: To develop a web server computer load control system of the IT company Civenty.

The explanatory note has had an analysis of existing monitoring systems, describing their main advantages and disadvantages. The research task was formulated with data from the analysis. After that, it was necessary to solve a scientific problem by building a model of web server load control system. The next step was to formulate technical requirements for the created system. The resulting structural scheme of the monitoring system has been built with these requirements. Next, software development was carried out with flow diagrams. In addition, the relationship between programs and their functionality has been described. The last step was to conduct an experiment with a web server load control system.

MONITORING, WEB SERVER, DOCKER, DOCKER-COMPOSE, CONTAINER, ANSIBLE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Веб-сервер – це сервер, який зберігає файли веб-сайту та відображає їх у браузері користувача;

Моніторинг – це постійний нагляд за процесом;

IT – Information Technology;

SSL – Secure Sockets Layer;

SSL-сертифікат – цифровий підпис для забезпечення зашифрованого безпечного мережевого трафіку;

YAML - YAML Ain’t Markup Language;

Docker-контейнер – це пакет програмного забезпечення, який має усі необхідні компоненти для запуску додатку у ізольованій середі;

ОС – операційна система;

ПЗ – програмне забезпечення;

БД – база даних;

AMQP - Advanced Message Queuing Protocol;

Telegraf-агент – це програмне забезпечення, яке встановлюється на досліджуваний сервер для збору та надсилання даних з нього;

SSD – Solid State Drive;

BASH - Born Again Shell;

JSON – JavaScript Object Notation.

ВСТУП

У наші часи, майже кожен представник бізнесу в нашій країні, від малого до великого, має зв’язок з комп’ютерними системами. І це не дивно, адже це зручно та надійно проводити операції у цифровому форматі, наприклад, продавати послуги через Інтернет, приймати участь у онлайн-аукціонах, робити замовлення чи розміщувати товар на сторінці власного інтернет-магазину.

Розуміючи усі ці переваги, велика кількість даних переноситься до власної комп’ютерної інфраструктури, за якою треба постійно доглядати. Тому, чим більше даних зберігається та оброблюється у комп’ютерній системі, тим більшу цінність несе безперервна працездатність. Адже кожна секунда непрацездатності системи може привести до збитків або зменшення прибутків, що може призвести до певних проблем для бізнесу. До того ж, величина збитків буде залежати від того, як швидко зможуть знайти проблему та виправити її завдяки черговим системним адміністраторам, які відповідальні за працездатність серверів.

Заради того, щоб зменшити тривалість часу на знаходження та вирішення проблеми або взагалі попередити її у комп’ютерних системах завантажують та встановлюють систему моніторингу. На сьогоднішній день, неможливо уявити існування будь-якої комп’ютерної інфраструктури без інструментів моніторингу працездатності. Адже за допомогою моніторингу, ми можемо дізнатися, коли може закінчитися вільне місце на накопичувачах, скільки зайнято оперативної пам’яті, на скільки процентів використовуються ядра процесору на даний час або як сильно навантажена комп’ютерна мережа та багато іншого. Про наявність кожної з існуючих проблем, черговий системний адміністратор може бути проінформований за допомогою панелі графічного інтерфейсу моніторингу або SMS-повідомленнями, яке надсилає дане програмне забезпечення. Тому, розуміючи як правильно налаштовувати

та оптимально використовувати функціонал системи моніторингу, то можливо буде реалізувати процес попереджування проблем, без проблем у працездатності.

У глобальній мережі є в наявності декілька десятків різних систем моніторингу. Вони можуть бути безкоштовними чи навпаки або з відкритим кодом чи з пропрієтарним кодом. Кожна система моніторингу має свої переваги та недоліки. Однак між ними є дещо спільне, а саме – обмеженість функціоналу, яка не дозволяє використовувати увесь функціонал програмного забезпечення та отримувати від цього усю користь. Це проявляється у тих випадках, коли конкретна система моніторингу не може збирати метрики про стан досліджуваного серверу для подальшого аналізу, які необхідні черговим системним адміністраторам. Ці обмеження можуть бути різними, наприклад інструмент моніторингу не може збирати дані про мережевий трафік або дане програмне забезпечення може працювати на пристроях тільки під керуванням операційної системи Windows.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є розробка та дослідження власної структури програмно-технічних засобів комп’ютерної системи контролю завантаженості веб-серверу ІТ-компанії Civenty.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- дослідити та проаналізувати найбільш застосовані системи моніторингу;
- на основі дослідження існуючих інструментів моніторингу, знайти їх переваги та недоліки;
- розробити модель системи контролю завантаженості веб-серверу;
- розробити структурну схему системи контролю завантаженості веб-серверу;
- розробити схеми алгоритмів для системи контролю завантаженості веб-серверу;

- програмно реалізувати моделі і алгоритми, реалізуючи системи контролю завантаженості веб-серверу;
- проведення експерименту системи контролю завантаженості веб-серверу для обґрунтування необхідності використання цієї системи, поміж конкурентів.

Об'єкт дослідження – існуюча система контролю завантаженості веб-серверу головного офісу IT-компанії Civenty.

Предмет дослідження - комп'ютерна система IT-компанії Civenty, яка забезпечує робочий процес працівників та функціонування веб-сайтів.

Методи дослідження. Для виконання отриманої цілі були застосовані методи аналізу та синтезу, порівняння, а також методу математичного моделювання.

Наукові положення:

1. Розроблені та обґрунтовані вимоги щодо моделювання системи контролю завантаженості веб-серверу.
2. Описаний зв'язок між Docker-контейнерами для забезпечення безперервної та надійної доставки даних з досліджуваного серверу на сервер моніторингу, який реалізується за допомогою використання YAML-синтаксису.

Наукові результати:

1. Одержані нові теоретичні відомості щодо створення математичних моделей для систем контролю завантаженості серверів, використовуючи Ланцюг Маркова.
2. Запропонований метод створення системи моніторингу серверів для контролю працездатності.
3. Обґрунтовано використання власної системи моніторингу веб-серверу поміж конкурентів завдяки виконанню порівняльних експериментальних досліджень.

Обґрунтованість і достовірність наукових положень, висновків та рекомендацій підтверджується тим, що в роботі було використано: сучасні методи для створення систем моніторингу досліджуваних серверів з використанням Docker-контейнерів, сучасне обладнання для побудови системи, найпоширенішими практиками керування Docker-контейнерами, ефективний набір інструментів для написання та керуванням програмного коду та проведення експериментальних досліджень.

Практичне значення отриманих результатів полягає у доцільності використання системи контролю завантаженості веб-серверу власної розробки, на основі отриманих експериментальних висновках.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Стан питання

На сьогоднішній день, кожна компанія, яка веде бізнес в Інтернеті має власний веб-сервер на якому розміщуються один веб-сайт чи декілька. За допомогою веб-сторінок продаються будь-які товари чи послуги у реальному часі. Тому, щоб бути конкурентоспроможними серед конкурентів, веб-сервер повинен працювати безперервно та забезпечувати стабільну роботу. Адже, кожна хвилина його простою або нестабільної роботи може призвести до втрати потенційних клієнтів, а це означає втрата прибутку.

Розуміючи масштаб можливих збитків, підприємець наймає системних інженерів для вирішення цього питання. Дані спеціалісти повинні будуть не тільки зробити початкове налаштування веб-сервер, але й забезпечити його подальшу безперебійну роботу.

Для контролю стану комп’ютерної системи використовують інструменти моніторингу. З їх допомогою системні інженери можуть слідкувати за працездатністю веб-серверу, переглядаючи дані у будь-якому вигляді в графічній панелі керування за допомогою веб-браузеру у реальному часі. Також, при наявності проблеми дане програмне забезпечення можуть інформувати чергових системних адміністраторів про наявність проблеми за допомогою листа по електронній пошті чи за допомогою SMS-повідомлення.

Однак, слід розуміти, що не кожен інструмент моніторингу може задовольняти усім потребам. Наприклад, деякі системи моніторингу можуть збирати лише ті дані, про які їй відомо, без можливості розширення чи оновлення. Або інструмент моніторингу має багатий функціонал, який може покрити усі потреби, але він споживає дуже багато ресурсів, що може сповільнювати роботу веб-сервера. Можуть навіть виникати ситуації, що необхідно надсилати повідомлення про проблеми в месенджер Telegram замість листа на електрону пошту.

Слід ще розуміти, що на серверах під керуванням різних операційних систем, можуть бути різні файли налаштувань тієї же самої системи моніторингу. А це вже унеможливлює виконання швидкого налаштування моніторингу на різних веб-серверах через пошук необхідних параметрів та значень для подальшого редагування.

Враховуючи усі ці проблеми, постає питання у тому, що вендори програмного забезпечення не можуть покрити будь-які примхи користувачів. І тому, споживачам потрібно йти на компроміси обмежуючи свої потреби, а це вже збільшує шанс на те, що може статися проблема, яка не буде виявлена системою моніторингу, що може привести до збитків для підприємця.

1.2 Аналіз існуючих систем моніторингу

Система моніторингу – це система чи набір систем, яка дозволяє побачити поточний стан комп’ютерної системи у реальному часі. Зазвичай, компонентами системи моніторингу є: метрики, агент, моніторинг, та система сповіщення.

Метрики – це необроблені дані, які мають в собі інформацію за певний проміжок часу для подальшого зберігання та аналізу [1]. Це може бути дані про навантаження будь-якого компонента серверу, які надаються операційною системою або будь-які інші дані, які можуть бути надані різними програмами. За допомогою метрик, ми можемо виконувати аналіз для того, щоб мати уяву про поточний стан комп’ютерної системи. Також ми можемо їх використовувати для створення прогнозів щодо майбутнього стану системи, заради того, щоб попередити та вже знешкодити проблему до початку її виникнення. Існують метрики наступних типів:

— метрики вузла – оцінюють використання та продуктивність обладнання;

— метрики додатків – оцінюють працездатність та навантаження на додаток;

— метрики мереж – оцінюють правильність роботи додатка у комп’ютерній мережі;

— метрики зовнішніх залежностей – оцінюють працездатність сервісів, з якими додаток має зв’язок.

Агент – це програмне забезпечення, яке збирає дані з системи в режимі реального часу та надсилає у вигляді метрики до моніторингу.

Моніторинг – це процес збору, зберігання, візуалізації та аналізу даних, які надані агентами для відображення поточного стану комп’ютерної системи. Тобто, за допомогою моніторингу ми можемо побачити оброблені дані з різних метрик, за допомогою яких, ми можемо отримати корисну для себе поточну інформацію. При цьому, дані можливо буде побачити у текстовому форматі або у вигляді графіків, які будуть відображатися в графічному інтерфейсі. На рисунку 1.1 зображено приклад оброблених даних моніторингом у вигляді графіків.

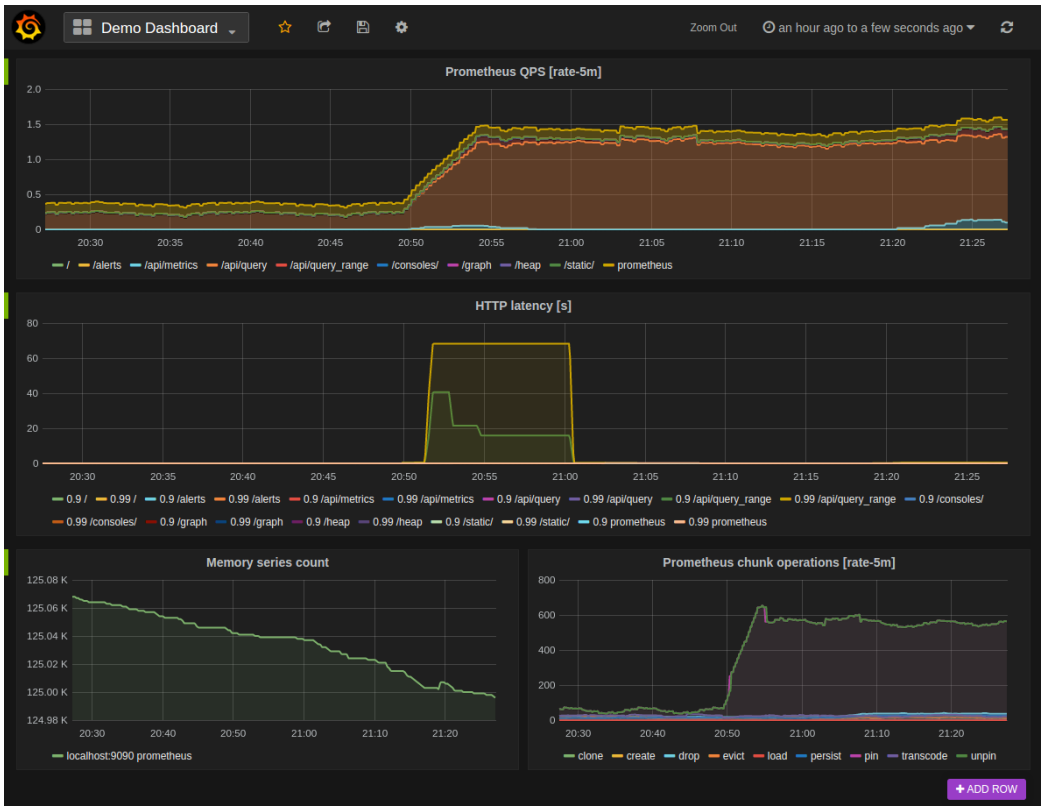

Рисунок 1.1 – Оброблені дані моніторингом у вигляді графіків

Система сповіщення – це один із компонентів системи моніторингу для виконання певних дій при отриманих метрик. При цьому, система сповіщення складається з двох компонентів:

— умова – це граничне значення, яке порівнюється з отриманими даними з метрик;

— дія – це здійснення певних кроків, які необхідно виконати, якщо дані з метрик не задовольняє умові.

За допомогою системи сповіщення, ми можемо реалізувати пасивний моніторинг, який дозволяє черговому системному адміністратору постійно не дивитися у графічний інтерфейс моніторингу, на відміну від активного моніторингу. І реагувати лише тільки в тому випадку, коли порушується умова.

Зазвичай, система сповіщення застосовується лише тоді, коли необхідно створити тригер на конкретну проблему. Коли створюється тригер на проблему, дуже важливо, правильно описати повідомлення, а саме, з чим зв’язана вона зв’язана. Це допоможе черговому системному адміністратору швидко зрозуміти проблему. Також можливо задати автоматичну дію, при порушення умови. Наприклад, якщо в кореневому розділі сервера під керуванням дистрибутиву CentOS залишилось 5 відсотків вільного місця, або менше, то необхідно видалити файли з директорії /tmp/.

У 2021 році існує вже понад декілька десятків різних систем моніторингу, але розглянемо серед них лише найвідоміші.

1.2.1 Система моніторингу Zabbix

Zabbix – це один з найпопулярніших систем моніторингу з відкритим кодом. Дане програмне забезпечення пропонує декілька варіантів моніторингу, а саме: мережі, сервери, хмари, бази даних, програми. Завдяки тому, що він безкоштовний у використанні, Zabbix має велику спільноту зі всього світу. Ця система моніторингу надає користувачам багато варіантів візуалізації даних. Ця система моніторингу підтримує Windows, UNIX та

UNIX-подібні операційні системи. Однією з цікавих особливостей Zabbix – є власна система прогнозування, яка може попередити про проблему, ще до її появи. Дана система моніторингу дозволяє створювати власні метрики поміж тих, що вже існують для збору та аналізу.

Переваги системи моніторингу Zabbix:

— безкоштовний з відкритим кодом;

— можливість швидкого знаходження пристроїв;

— інформативна документація;

— велика спільнота зі всього світу;

— швидка та проста взаємодія з додатками використовуючи Zabbix API.

Недоліки системи моніторингу Zabbix:

— споживає багато ресурсів;

— важкий у налаштуванні;

— перенасичений графічний інтерфейс.

1.2.2 Система моніторингу Nagios

Nagios – це програмне забезпечення з відкритим кодом, яке дозволяє налаштовувати моніторинг майже будь-яких компонентів, наприклад: мережевих протоколів, системи, додатків, веб-серверів, веб-сайтів, тощо. Такий функціонал досягається завдяки модульній архітектурі [2]. Тобто, архітектура Nagios може бути розширена за допомогою плагінів.

Плагіни – це спеціалізоване програмне забезпечення, які викликаються сервером моніторингу для збору та звітування про дані у реальному часі та подальша відправка до серверу.

Існує три версії Nagios:

– Nagios Core – безкоштовна, але має обмежений функціонал;

– Nagios Standard – вартість ліцензії становить 1995 доларів, необмежений функціонал тільки на 100 вузлів;

− Nagios Standard – вартість ліцензії становить 3495 доларів, необмежений функціонал.

Переваги системи моніторингу Nagios:

− використання модульної архітектури;
− програмне забезпечення з відкритим кодом;
− простий у використанні інтерфейс;
− можливість автоматичного вирішення проблем.

Недоліки системи моніторингу Nagios:

— обмежений функціонал у безкоштовній версії;
— підтримка лише UNIX та UNIX-подібних операційних систем;
— важкий у налаштуванні.

1.2.3 Система моніторингу Munin

Munin – це програмне забезпечення з відкритим кодом для проведення моніторингу комп'ютерних систем, мереж та усіх складових комп'ютерної інфраструктури. Даний інструмент моніторингу простий у налаштуванні та легко розширює свій функціонал за допомогою плагінів [3]. Наразі існують, більш ніж 500 плагінів. Також можливо написати власні плагіни на будь-якій мові програмування. Нажаль, дана система моніторингу призначена лише для активного моніторингу, так як в ній відсутня система сповіщення.

Переваги системи моніторингу Munin:

— безкоштовний з відкритим кодом;
— простий у використанні інтерфейс;
— багатий функціонал за допомогою плагінів;
— можливість створення власних плагінів;
— простий у налаштуванні.

Недоліки системи моніторингу Munin:

— відсутність системи сповіщення;

— використовує багато ресурсів через створення надмірної кількості потоків;

— неможливість модифікації графіків.

1.2.4 Система моніторингу Cacti

Cacti – це програмне забезпечення з відкритим кодом для проведення моніторингу комп’ютерних мереж та зображення отриманих даних. Він може бути встановлений на сервер під керуванням будь-яких операційних систем. Дана система моніторингу може розширювати свій функціонал за допомогою плагінів, серед яких є плагіни для системи сповіщення, що означає можливість використання його для пасивного моніторингу [4]. Однак, на відміну від деяких конкурентів, Cacti не підтримує використання плагінів, які написані користувачами, тому можна вважати обмеженим.

Переваги системи моніторингу Cacti:

— безкоштовний з відкритим кодом;
— простий у використанні інтерфейс;
— можливість використання плагінів для розширення функціоналу;
— підтримка усіх операційних систем;
— зручний графічний інтерфейс;
— легкий у налаштуванні.

Недоліки системи моніторингу Cacti:

— обмежений функціонал;
— після перезавантаження Cacti пропадають зібрані дані;
— важка у розумінні документація;
— неактивна спільнота.

1.2.5 Система моніторингу Icinga

Icinga – це безкоштовне програмне забезпечення з відкритим кодом для проведення моніторингу комп’ютерних систем та мереж. Дана система моніторингу підтримує усі існуючі операційні системи. Icinga має активну

спільноту зі всього світу, яка допомагає покращувати цей продукт. Він має широкий функціонал системи сповіщень, так як він дозволяє відправляти повідомлення про проблему не тільки у вигляді листа на електрону пошту чи SMS-повідомлення на мобільний телефон, але й ще дозволяє надсилати повідомлення до месенджерів. Icinga має широкий функціонал для налаштування графічного зображення графіків з даними, а також веб-інтерфейсу. Також можливо налаштувати систему моніторингу Icinga, використовуючи командну строку.

Переваги системи моніторингу Icinga:

— безкоштовний з відкритим кодом;

— функціональна система сповіщення;

— широкі можливості налаштування веб-інтерфейсу та графіків;

— активна спільнота зі всього світу, яка постійно займається оновленням.

Недоліки системи моніторингу Icinga:

— важкий у налаштуванні;

— оновленнями системи моніторингу займається лише активна спільнота.

1.3 Проблеми сучасних систем моніторингу

Згідно з проаналізованих матеріалів, ми побачили, що найчастішими проблемами сучасних систем моніторингу є:

— обмежений функціонал;

— обмежені налаштування графічного інтерфейсу;

— складність налаштовувань;

— споживання великої кількості системних ресурсів;

— обмежена підтримка операційних систем.

1.4 Постановка завдання дослідження

Проаналізувавши необхідність використання бізнесом системи моніторингу, ми можемо сформувати наступні задачі для кваліфікаційної роботи:

— дослідити переваги та недоліки сучасних систем моніторингу;

— визначити інструменти для побудови власної системи моніторингу для IT компанії Civenty;

— визначити принцип роботи нової системи моніторингу;

— обґрунтувати доцільність використання власної системи моніторингу.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Загальна характеристика комп’ютерної системи

IT-компанія Civenty була заснована у 2015 році. Ця компанія надає послуги провайдера високоякісних бізнес-рішень для клієнтів зі всього світу [5]. Поміж цього, Civenty розроблює та підтримує існуючі веб-проекти та додатки, проводить експертизи існуючих проектів, заради покращення роботи процесів у комп’ютерних системах. Команда розробників використовує наступні сучасні мови програмування для створення проектів, такі як:

- PHP;
- Python;
- Java;
- JavaScript.

Об’єктом дослідження є головний офіс IT-компанії Civenty, який знаходиться на першому поверсі житлового комплексу “Панорама”. Даний офіс надає робочі місця для 66 працівників, які оснащені персональними комп’ютерами з встановленим програмним забезпеченням, монітором, клавіатурою, комп’ютерною мишею та навушниками. В залежності від наданих задач розробнику його робоче місце може бути оснащеним додатковою периферією.

Комп’ютерна мережа IT-компанії Civenty складається з:

— 70 робочих станцій;
— 3 комутаторів;
— 1 маршрутизатору;
— 1 веб-сервер;
— 2 принтерів.

Офіс може бути використаним цілодобово. Працівники можуть прийти у будь-який час до свого робочого місця задля виконання професійних обов’язків.

2.2 Структура об’єкту дослідження

Головний офіс IT-компанії Civenty має наступні складові комп’ютерної системи, такі як:

- маршрутизатор;
- комутатори;
- персональні комп’ютери;
- веб-сервер;
- принтери.

Саме ці компоненти забезпечують стабільну роботу найманців та своєчасно надавати послуги своїм клієнтам. На рисунку 2.1 зображено розташування компонентів комп’ютерної системи в офісі IT-компанії.

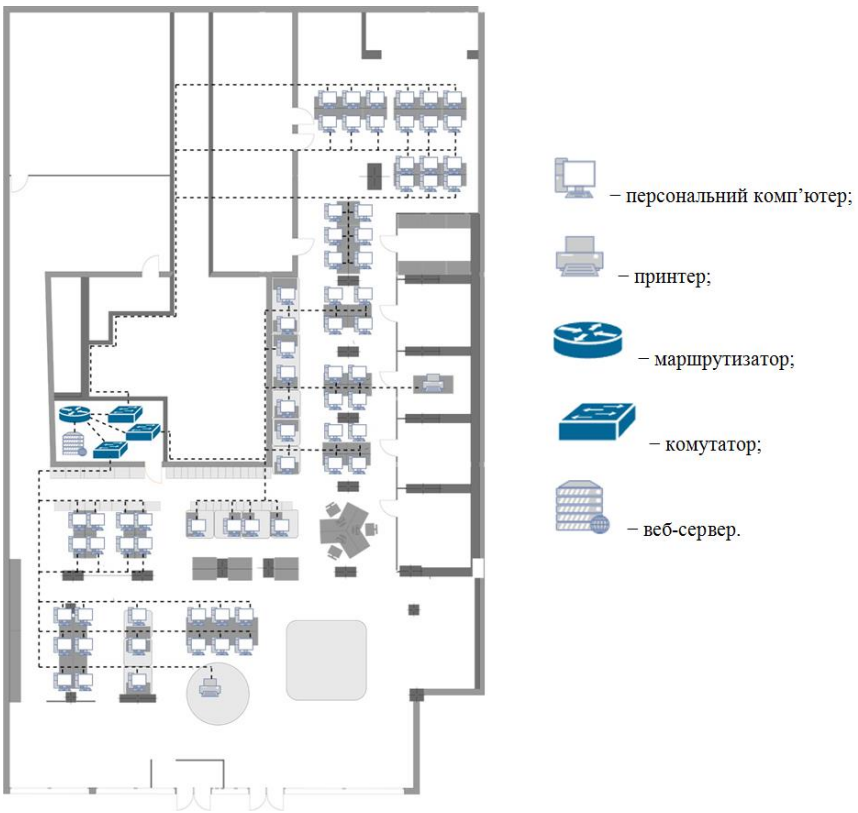

Рисунок 2.1 – Схема розміщення компонентів комп’ютерної системи в офісі

2.3 Обґрунтування і вибір методів дослідження

При виборі компонентів для побудови удосконаленої системи моніторингу були проаналізовані різні методи. Дослідивши кожну з них, було вирішено обрати наступні методи:

— метод аналізу та синтезу;

— метод порівняння;

— метод математичного моделювання.

2.3.1 Методи аналізу та синтезу

Сучасні системи моніторингу мають різні недоліки, в залежності від обраного вендору користувачем. До найпоширеніших проблем відносяться:

— обмежений функціонал – програмне забезпечення не може збирати певні метрики від веб-серверу, який аналізується. Наприклад, системи моніторингу не можуть збирати дані про тривалість SSL-сертифікату веб-сайту, який знаходиться на цьому сервері. Або взагалі відсутня можливість додавати власні скрипти для збору необхідних даних;

— обмежені налаштування графічного інтерфейсу – деякі системи моніторингу не дозволяють редагувати зовнішність її головної сторінки. А це означає, що ми не можемо винести необхідні графіки з даними на головну сторінку, щоб виділити поміж тих графіків, які взагалі можуть бути не цікавими;

— складність налаштовувань – через перенасиченість деяких сучасних систем моніторингу дуже важко зробити певні налаштування при її встановленні на сервер. А це може зайняти багато часу або призвести до зайвих помилок;

— споживання великої кількості системних ресурсів – такі потужні системи моніторингу, як Zabbix можуть задовільнити усі потреби системних інженерів, через велику кількість метрик, який він може збирати та аналізувати при його встановленні. Однак, він може знаходити й такі метрики,

які можуть взагалі бути не цікавими користувачу, наприклад, відсоток кеш влучання бази даних. Тому ці зайві елементи даних можуть використовувати ресурси веб-серверу. За замовчуванням, система моніторингу Zabbix встановлює близько 200 елементів даних для збору метрик з комп'ютерного вузла [6]. Як наслідок, треба перевіряти, чи всі вони взагалі потрібні. Адже, зайві елементи даних можуть споживати у надлишковій кількості;

— обмежена підтримка операційних систем – деякі системи моніторингу можуть не підтримувати UNIX-подібні операційні системи чи ОС Windows. Цей недолік є серйозним, через те, що в залежності від потреб клієнта можливе використання серверів з різними операційними системами.

При використанні власної системи моніторингу можливою проблемою може бути те, що агент Telegraf може не збирати певні метрики за допомогою існуючих плагінів. Однак, ця проблема легко вирішується створенням власного плагіну для збору даних з комп'ютерного вузла.

Як наслідок, при встановленні власної системи моніторингу вирішуються вищезгадані проблеми сучасних систем моніторингу, що дозволить досліджувати лише ті дані, які нам необхідні. А також, ми можемо створити власні плагіни, які б дозволили збирати лише ті метрики, які цікаві тільки ІТ-компанії Civenty.

2.3.2 Порівняльний аналіз існуючих систем моніторингу

Порівняємо п'ять найпопулярніших систем моніторингу з власною розробкою.

2.3.2.1 Переваги та недоліки системи моніторингу Zabbix

Переваги системи моніторингу Zabbix:

— безкоштовний з відкритим кодом;

— можливість швидкого знаходження пристроїв;

— інформативна документація;

— велика спільнота зі всього світу;

— швидка та проста взаємодія з додатками використовуючи Zabbix API.

Недоліки системи моніторингу Zabbix:

— споживає багато ресурсів;

— важкий у налаштуванні;

— перенасичений графічний інтерфейс.

2.3.2.2 Переваги та недоліки системи моніторингу Nagios

Переваги системи моніторингу Nagios:

— використання модульної архітектури;

— програмне забезпечення з відкритим кодом;

— простий у використанні інтерфейс;

— можливість автоматичного вирішення проблем.

Недоліки системи моніторингу Nagios:

— обмежений функціонал у безкоштовній версії;

— підтримка лише UNIX та UNIX-подібних операційних систем;

— важкий у налаштуванні.

2.3.2.3 Переваги та недоліки системи моніторингу Munin

Переваги системи моніторингу Munin:

— безкоштовний з відкритим кодом;

— простий у використанні інтерфейс;

— багатий функціонал за допомогою плагінів;

— можливість створення власних плагінів;

— простий у налаштуванні.

Недоліки системи моніторингу Munin:

— відсутність системи сповіщення;

— використовує багато ресурсів через створення надмірної кількості потоків;

— неможливість модифікації графіків.

2.3.2.4 Переваги та недоліки системи моніторингу Cacti

Переваги системи моніторингу Cacti:

— безкоштовний з відкритим кодом;
— простий у використанні інтерфейс;
— можливість використання плагінів для розширення функціоналу;
— підтримка усіх операційних систем;
— зручний графічний інтерфейс;
— легкий у налаштуванні.

Недоліки системи моніторингу Cacti:

— обмежений функціонал;
— після перезавантаження Cacti пропадають зібрані дані;
— важка у розумінні документація;
— неактивна спільнота.

2.3.2.5 Переваги та недоліки системи моніторингу Icinga

Переваги системи моніторингу Icinga:

— безкоштовний з відкритим кодом;
— функціональна система сповіщення;
— широкі можливості налаштування веб-інтерфейсу та графіків;
— активна спільнота зі всього світу, яка постійно займається оновленням.

Недоліки системи моніторингу Icinga:

— важкий у налаштуванні;
— оновленнями системи моніторингу займається лише активна спільнота.

2.3.2.6 Переваги та недоліки власної системи моніторингу

Переваги власної системи моніторингу:

— створення власних плагінів для збору різних метрик;

— широкі можливості для створення власних графіків;
— надсилання повідомлень про проблеми у месенджер;
— раціональне використання ресурсів веб-серверу;
— легкий у налаштуванні.

Недоліки власної системи моніторингу:

— необхідність власноруч підтримувати систему моніторингу.

Проаналізувавши сучасні системи моніторингу, можливо зробити висновок, що кожна з них не може задовольняти усі потребам користувачів. Споживачі повинні змиритися з існуючими недоліками, щоб встановити дане програмне забезпечення. Розроблена власна система моніторингу дозволяє робити моніторинг веб-серверу без вищезгаданих недоліків. Також, надаючи можливість для впровадження власних плагінів для збору даних про різні складові серверу та й не тільки. До того ж, можливе редагування відображення існуючих графіків з даними, а також головної сторінки системи моніторингу для відображення даних з високим пріоритетом.

2.3.3 Загальна модель системи моніторингу

Сучасні системи моніторингу, як і система контролю завантаженості повинні забезпечувати постійний контроль за станом досліджуваних об’єктів заради виявлення несправності або зменшення їх працездатності. З їх допомогою, чергові системні адміністратори можуть запобігти майбутнім проблемам.

Математичне модель – це опис певного явища з використанням математичних моделей [7]. Метою створення математичної моделі є виявлення несправностей з використанням моніторингу. Об’єктом моделювання є моніторинг веб-серверу.

Однак, перш ніж виконати математичне моделювання системи моніторингу, необхідно ознайомитися з її процесами. Зазвичай процес моніторингу складається з двох частин: збір метрик з досліджуваного об’єкту

та аналіз отриманих даних. Для збору даних з досліджуваного серверу використовується агент, який збирає дані про використання ресурсів та за запитом серверу моніторингу надсилає до нього. Після цього, отримані дані аналізуються для прийняття рішення чи є сервер справним чи ні. Якщо система моніторингу побачить, що існує проблема, то спрацює тригер, який відправить повідомлення черговому системному адміністратору про існуючу проблему. Для вирішення цієї задачі, використовується формула 2.1, у якій використовуються наступні зміні:

— C – загальна множина отриманих метрик з веб-серверу;
— $C_п$ – підмножина значень, що не змушують спрацювати тригер;
— $C_н$ – підмножина значень, що змушують спрацювати тригер.

$$C = \{C_п, C_н\} \quad (2.1)$$

Так як, події під час моніторингу системи мають випадковий характер, тому для їх вивчення слід використовувати ймовірнісні математичні моделі теорії масового обслуговування. Для цієї задачі, найкраще підходить ланцюг Маркова. Ланцюг Маркова – це послідовність випадкових дій, де ймовірність майбутньої події, залежить від попередніх станів [8]. Ланцюг має вигляд графа з вершинами та ребрами. Де вершинами є стан системи, а ребрами є інтенсивність переходу з одного стану до іншого. За допомогою графу можливо знайти ймовірність кожного з станів, у випадку зміни параметрів з часом.

Вхідними даними для моделювання будуть стани досліджуваного серверу. Нижче наведені стани працездатності веб-серверу, який досліджуємо:

— S_0 – відсутність несправностей;
— S_1 – проблеми в роботі накопичувача;
— S_2 – проблеми в роботі оперативної пам’яті;

— S_3 – проблеми в роботі процесору;

— S_4 – фізично несправний;

— S_5 – непрацездатна система.

Описаний ланцюг у вигляді графу подій під час моніторингу досліджуваного веб-серверу зображено на рисунку 2.2.

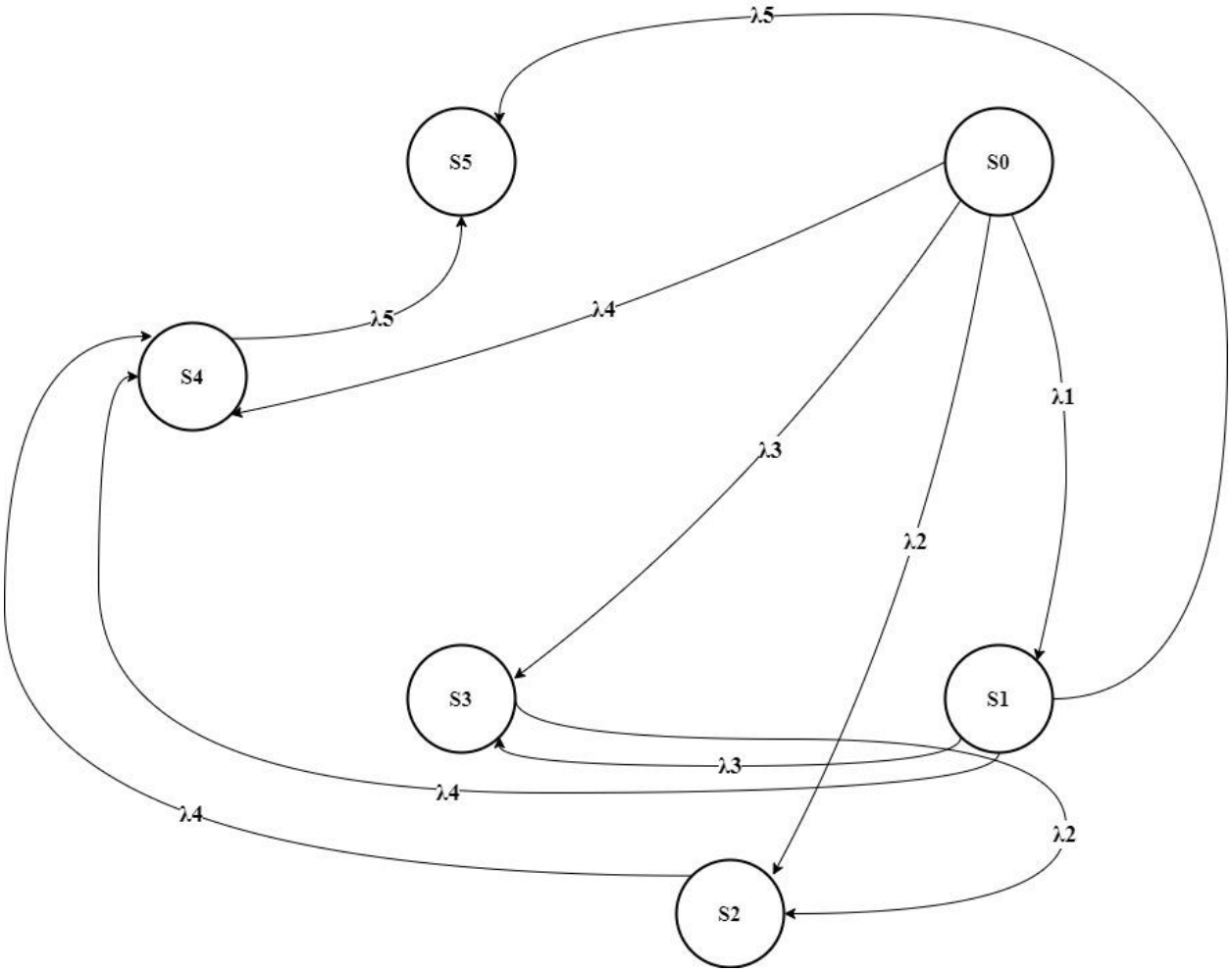

Рисунок 2.2 – Граф подій під час моніторингу досліджуваного веб-серверу

Після етапу опису вхідних даних необхідно зробити математичний опис моделі. Тому, будемо вважати, що будь-яка подія несправності є випадковим процесом з обмеженим числом станів. Кожна подія відбувається по-одиноко, тому дана модель буде одинарною. До того ж, кількість станів, які надійшли не містять післядій для наступного. Ймовірність *i*-го стану визначається, як

ймовірність знаходження у S_i стані. Тобто це є ймовірністю знаходження *i*-го несправного компонента веб-сервера системою моніторингу. Під час роботи моніторингу запити до агента, який збирає метрики з заданим пріоритетом, який назначений системним адміністратором. Тому, чим вище буде пріоритет, тим більшою буде інтенсивність запиту. У якості показника інтенсивності запиту, використовується λ_i, яка показує інтенсивність запиту *i*-го досліджуваного компоненту веб-сервера.

Опис роботи системи моніторингу виглядає наступним чином, шо зі стану S_0, який виконує запит до агента про певний компонент досліджуваного веб-серверу, при виявленні проблеми переходить до стану S_i з інтенсивністю запиту λ_i та повідомляє системному адміністратору про наявність проблеми. Враховуючи цей опис роботи та граф подій під час моніторингу досліджуваного веб-серверу, можемо написати математичну модель усього процесу моніторингу веб-серверу, застосовуючи систему рівнянь 2.2:

$$\begin{cases} \frac{dp_0}{dt} = -p_0 * (\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4), \\ \frac{dp_1}{dt} = p_0 * \lambda_1 - p_1 * (\lambda_3 + \lambda_4 + \lambda_5), \\ \frac{dp_2}{dt} = p_0 * \lambda_2 + p_3 * \lambda_2 - p_2 * \lambda_4, \\ \frac{dp_3}{dt} = p_0 * \lambda_3 + p_1 * \lambda_3 - p_3 * \lambda_2, \\ \frac{dp_4}{dt} = p_0 * \lambda_4 + p_1 * \lambda_4 + p_2 * \lambda_4 - p_4 * \lambda_5, \\ \frac{dp_5}{dt} = p_1 * \lambda_5 + p_4 * \lambda_5. \end{cases} \quad (2.2)$$

Рішення цієї системи рівнянь дозволить зрозуміти процес виявлення несправних компонентів веб-серверу завдяки відслідковуванням ймовірностей у певний інтервал часу.

У випадку виявлення ймовірнісного стану системи моніторингу загалом у певному стані, треба використати лінійну систему рівнянь, прирівнявши ліві частини рівнянь до нулів. При моделюванні майбутнього стану системи

необхідно враховувати попередній крок. Тому ймовірний майбутній стан системи моніторингу можна описати формулою 2.3:

$$p(AB) = p(A) * p(B|A), \qquad (2.3)$$

де *p(AB)* – це ймовірність подій А та В; *p(A)* – ймовірність події А; *p(B|A)* – ймовірність події В, якщо буде подія А.

Враховуючи ймовірності цих станів, ймовірність виникнення стану S_n зображено на формулі 2.4:

$$p(S_0 S_2 \dots S_n) = p(S_0) \prod_{j=1}^{n} p\left(S_j \left| S_0 S_1 \dots S_{j-1}\right.\right) \qquad (2.4)$$

Виходячи з формули 2.4, процес є n-зв'язним, так як на результат впливає результат *n*-попередніх випробувань.

У випадку з системою моніторингу, яка буде розроблена, спираючись на граф подій під час моніторингу досліджуваного веб-серверу, який має шість станів (S_0, S_1, S_2, S_3, S_4, S_5), слід врахувати, що довести до стану непрацездатності можуть *n-1* попередні випробування. Тому, спираючись на цю інформацію, справедливою є формула 2.5:

$$p(S_1 \dots S_n) = p(S_1) \prod_{j=2}^{n} p\left(S_j \left| S_1 S_2 \dots S_{j-1}\right.\right) \qquad (2.5)$$

Завдяки отриманій формулі, можемо описати процедуру прогнозування системи контролю завантаженості веб-серверу до стану непрацездатності на основі попередніх отриманих даних, виявивши проблемний компонент.

2.3.4 Створення моделі моніторингу в КС

Перед розробкою програмного забезпечення та проведення експериментів для системи контролю завантаженості веб-серверу необхідно

розробити модель моніторингу в КС, яка б відповідала клієнт-серверній архітектурі. Для побудови моделі буде застосовано середовище MATLAB, у якій встановлений Simulink. Simulink – це середовище для візуалізації імітаційного моделювання, яке використовує блок-діаграми для побудови динамічних моделей [9]. Головними перевагами у використанні цього ПЗ є: велика кількість інструментів для аналізу та проектування з допомогою використання бібліотек; взаємодія з мовами програмування C/C++ та повна інтеграція з MATLAB.

Постановкою задачі для моделювання є: спираючись на попередньо отриману інформацію про існуючі системи моніторингу, необхідно створити модель моніторингу КС.

Модель моніторингу КС зображено на рисунку 2.3.

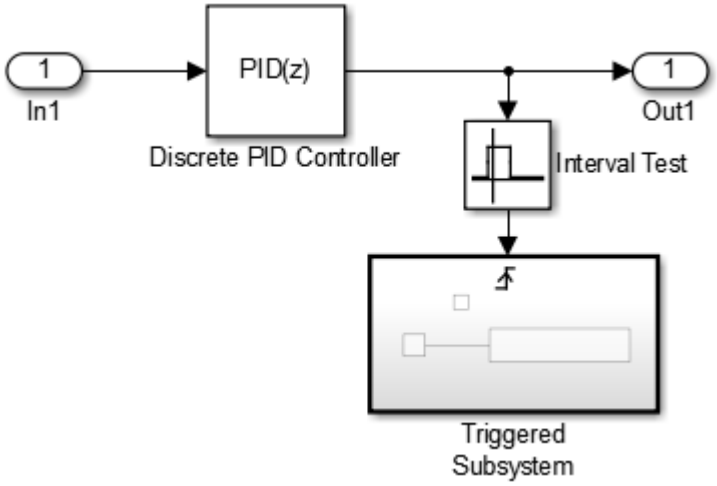

Рисунок 2.3 – Модель моніторингу в КС

Модель системи моніторингу в КС має наступні блоки:

— In1 – вхід системи;

— Discrete PID Controller – перевіряє отримані метрики залежних від часу отриманих з веб-серверу. В залежності від того, які дані отримав контролер він може їх надіслати до блоку Out1, якщо немає проблем, в іншому випадку надсилається сигнал до Interval Test;

— Interval Test – перевіряє чи знаходиться значення у заданому діапазоні;

— Triggered Subsystem – підсистема, яка виконується при отриманні керуючого сигналу від блоку Interval Test;

— Out1 – вихід системи.

Підсистема Triggered Subsystem зображена на рисунку 2.4.

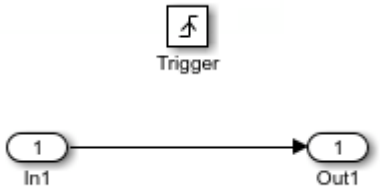

Рисунок 2.4 − Підсистема Triggered Subsystem

Підсистема має наступні складові:

— блок In1 – вхід підсистеми;

— блок Out1 – вихід підсистеми.

В залежності від отриманого сигналів з блоків Interval Test та Discrete PID Controller, Triggered Subsystem може змінювати сигнал при виході з блоку Out1.

2.4 Висновки до розділу

У даному розділі була описана загальна характеристика комп’ютерної системи підприємства. До того ж, було виявлено об’єкт дослідження та описано його структуру. Після цього були розглянуті методи дослідження для аналізу сучасних систем моніторингу та побудови власної удосконаленої системи. Для цієї роботи було виявлено, що найкраще підходять наступні методи: аналізу та синтезу; порівняння та математичного моделювання. Завдяки методу аналізу та синтезу були виявлені проблеми сучасних систем моніторингу, які необхідно враховувати при побудові власної системи.

Спираючись на метод порівняння були проаналізовані переваги та недоліки найпопулярніших систем моніторингу. Це дозволило врахувати та застосувати переваги кожної з цих систем та уникнути більшості недоліків. За допомогою методу математичного моделювання було описано процедуру системи контролю завантаженості веб-серверу.

3 СИНТЕЗ СИСТЕМИ КОНТРОЛЮ ЗАВАНТАЖЕНОСТІ ВЕБ-СЕРВЕРУ

3.1 Цілі впровадження системи

Перед впровадженням системи контролю завантаженості веб-серверу задано наступні цілі:

— контроль працездатності досліджуваного серверу;

— наявність інструментів для аналізу поточного стану апаратних компонентів досліджуваного веб-серверу;

— надсилання повідомлень про стан компонентів досліджуваного серверу за заданими контактами чергових системних адміністраторів;

— зберігання отриманих метрик про завантаженість компонентів досліджуваного веб-серверу у базі даних;

— зменшення часу на пошук та вирішення можливих проблем веб-серверу;

— попередити наявність майбутніх проблем досліджуваного серверу.

3.2 Формулювання технічних вимог до системи контролю завантаженості веб-серверу

3.2.1 Вимоги до системи в цілому

3.2.1.1 Вимоги до реалізації системи

Сфера застосування системи контролю завантаженості веб-серверу – сервер, який належить IT-компанії у якості платформи для розробки та тестування розподіленого та мігруючого програмного забезпечення, працюючий під керуванням ОС CentOS 7. Система має бути реалізована з використанням Docker-контейнерів. Так як, одним із головних недоліків

сучасних систем моніторингу – це надмірне використання ресурсів серверу. Використання Docker-контейнерів, дозволяє контролювати споживання на вузлі, якому вони працюють. Для керування контейнерами має бути використано Docker-Compose. Його функціоналу достатньо для вирішення поставлених задач. Для керування контейнерами Docker-Compose використовує синтаксис YAML(YAML Ain't Markup Language) [10], тому при написанні програми, розробник має розуміти даний синтаксис.

Для налаштування серверу, на якому буде встановлено систему контролю завантаженості веб-серверу, має бути використано Ansible. Дане ПЗ дозволить замінити ручне налаштування серверу, використовуючи програму, яка написана використовуючи синтаксис YAML. Дане рішення дозволить налаштувати сервер моніторингу за декілька хвилин, замість декількох годин та зменшити ймовірність помилки при налаштуванні через людський фактор.

Для написання програми агенту, який буде збирати дані про використання ресурсів веб-серверу має бути використана мова програмування bash. Так як, досліджуваний сервер працює під керуванням операційної системи CentOS 7, а також функціонал даної мови програмування є достатнім для вирішення поставленої мети, тому немає необхідні у використанні додаткових бібліотек. Агент має працювати усередині Docker-контейнеру, щоб контролювати використання ресурсів.

Розробник повинен мати досвід роботи зі створенням та налаштуванням контейнерів, а також побудови зв'язку між ними. Перед початком розробки програми програміст повинен створити схему алгоритму ПЗ.

3.2.1.2 Вимоги до функцій виконуваних системою

Система контролю завантаженості веб-серверу повинна працювати постійно. Дана система повинна мати наступні функції:

— надавати можливості для збору метрик досліджуваного веб-серверу ІТ-компанії;

— графічний інтерфейс для аналізу отриманих даних;

— створення графіків для візуалізації метрик про завантаженість процесору, оперативної пам’яті та накопичувача досліджуваного серверу;

— створення тригерів на події при перевантаженні досліджуваних компонентів серверу.

3.2.2 Вимоги до видів забезпечення

Для реалізації роботи системи необхідним є встановлення програмних інструментів, таких як Docker та Docker-Compose на сервері під керуванням UNIX чи UNIX-подібної (CentOS, Debian, Fedora, RHEL, SLES, Ubuntu) операційної системи, які є необхідними для створення ПЗ. До того ж, ці інструменти не мають підтримки повного функціоналу на ОС Windows. Для стабільної роботи програмного забезпечення системи контролю завантаженості веб-серверу, майбутній сервер моніторингу повинен відповідати вимогам, які будуть достатніми для функціонування операційної системи та чотирьох Docker-контейнерів, згідно з рекомендованими системними потребами [11]:

— процесор має мати, не менш ніж 4 ядра;

— оперативна пам’ять з об’ємом, не менш ніж 8 ГБ;

— зарезервованого місця на накопичувачі для Docker-контейнерів та створеними ними лог-повідомлень, не менш чим 100 ГБ.

Також слід зазначити, щоб сервер мав встановлену одну з підтримуваних UNIX чи UNIX-подібних операційних систем з наступними версіями. Це необхідно для підтримки стабільної працездатності роботи Docker-контейнерів, згідно з офіційними системними потребами [12]:

— CentOS: 7, 8;

— Debian: 10, 11;

— Fedora: 33, 34, 35;

— RHEL: 7, 8;

— SLES: 15-SP2, 15-SP3;

— Ubuntu: 18.04, 20.04, 21.04, 21.10.

3.2.3 Вимоги до захисту інформації

Для захисту інформації, яка зберігається у системі контролю завантаженості веб-серверу необхідно ввести аутентифікацію користувачів за допомогою приватного та публічного ключа, замість використання логіну та паролю. Це дозволить захиститися від крадіжки паролю використовуючи метод брутфорс. Ключі мають бути зашифровані з використанням наступних алгоритмів: dsa, rsa, ecdsa, ed25519. Довжина rsa-ключа має бути, не меншою ніж 4096 бітів, згідно з рекомендаціями National Security Agency [13]. Для інших ключів, достатньо використовувати довжину ключа за замовчуванням. Також потрібно зашифрувати розділи накопичувача від витоку даних.

Необхідно зачинити порти на сервері моніторингу, які є не зашифрованими та ті, які не використовуються взагалі. Це дозволить зменшити вірогідність злому, використовуючи методи сканування мережевих портів.

Доступ до системи контролю завантаженості веб-серверу має бути лише у чергових адміністраторів. Інші користувачі не повинні мати доступ до серверу.

Далі необхідно створити процес щоденного створення бекапів даних про завантаженість веб-серверу з таблиць БД. Ці резервні повинні зберігатися на протязі 1 місяця. Видалення архівів з бекапами відбувається за допомогою системного пакету logrotate, який встановлюється разом з будь-якою UNIX чи UNIX-подібної операційної системи. Доступ до цих файлів повинен мати лише суперкористувач операційної системи. Це дозволить захистити чутливі резервні дані від сторонніх осіб.

Доступ до облікового запису суперкористувача має бути лише у провідного системного адміністратора. Для захисту цього аккаунту має бути використаний пароль довжиною не менш 12 різних символів, різними

регістрами та не повинні мати існуючі слова згідно рекомендаціям від Google [14].

3.2.4 Вимоги до якості реалізації функцій системи

Агент зі збору метрик повинен забезпечити:

— збір метрик про завантаженість процесору, оперативної пам’яті, накопичувача досліджуваного веб-серверу кожні 30 секунд;

— надсилання зібраних метрик про завантаженість досліджуваного серверу кожну хвилину.

Сервер моніторингу повинен забезпечити:

— збір метрик про завантаженість компонентів досліджуваного серверу, які надані агентом кожну хвилину;

— гарантовану доставку метрик з агенту до бази даних;

— зберігання метрик про завантаженість процесору, оперативної пам’яті, накопичувача досліджуваного веб-серверу на протязі 1 місяця;

— наявність інструментів для аналізу отриманих метрик з веб-серверу.

3.2.5 Вимоги до ергономіки системи

Складові системи контролю завантаженості веб-серверу повинні знаходитися у будівлі, де знаходиться ІТ-компанії у спеціальній кімнаті. Доступ до цієї кімнати повинен бути лише у системних адміністраторів. До того ж, ці складові повинні знаходитися у місцях, які зручні для обслуговування.

3.2.6 Вимоги до персоналу, який обслуговує систему

До персоналу, який обслуговує систему повинні відноситися два адміністратора систем, які мають академічну степінь бакалавра або вище. Періодичний інструктаж щодо техніки безпеки – 1 раз на рік.

Режим роботи − 1 зміна по 8 годин. Тривалість робочого місяця – від 28 до 31 діб, в залежності від календарного місяця.

3.3 Розробка схеми функціональної структури

Наступними функціями системи контролю завантаженості веб-серверу є:

— збір метрик про завантаженість процесору, оперативної пам’яті та накопичувача веб-серверу;

— зберігання та обробка метрик отриманих від веб-серверу;

— відображення отриманих метрик у графічному вигляді.

Вищезгадані функції виконуються на сервері моніторингу.

Враховуючи ці дані, маємо змогу побудувати схему функціональної структури, яка зображена на рисунку 3.1.

Система контролю завантаженості веб-серверу буде складатися з наступних компонентів, які будуть встановлені до серверу моніторингу: агент для збору метрик, ПЗ для гарантійної доставки даних з веб-серверу, база даних, графічний інтерфейс.

У якості агента було обрано Telegraf – це серверний агент з відкритим вихідним кодом, який встановлюється на досліджуваному об’єкті для збору метрик з сенсорів та систем [15]. Дане програмне забезпечення має великий функціонал для збору будь-яких даних. Це досягається завдяки використанню плагінів. Їх можна завантажити з репозиторію або написати власноруч. Тому, для кожного типу метрик необхідно встановити один плагін. На сьогоднішній день, Telegraf має 300 плагінів для збору різних категорій, таких як: лог-повідомлення, мережеві пристрої, системні ресурси, IoT-пристрої та інші. Завдяки цій варіативності, Telegraf не має конкурентів, серед програмного забезпечення для збору метрик.

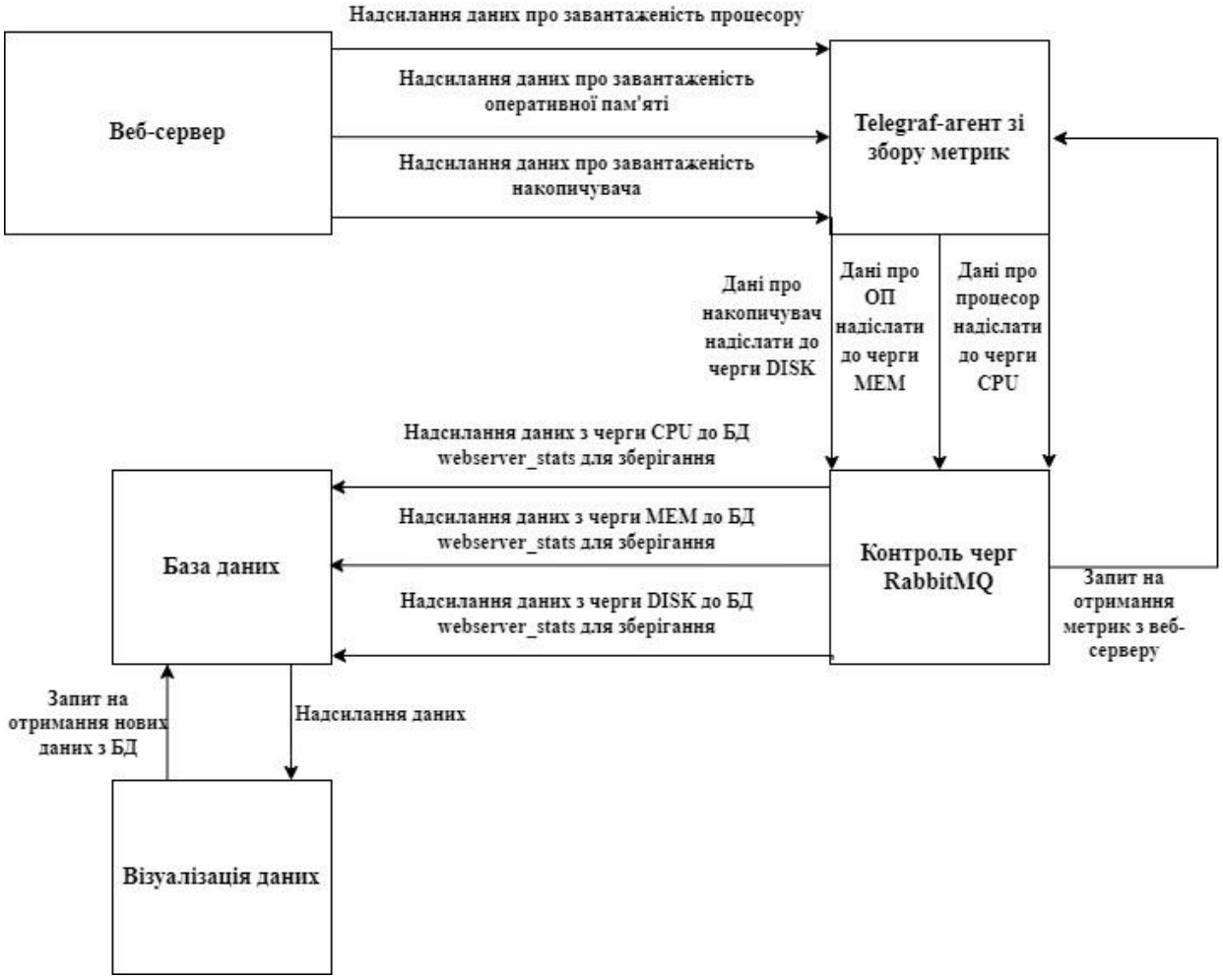

Рисунок 3.1 – Схема функціональної структури

Після того, як Telegraf-агент зібрав дані, їх необхідно відправити до бази даних. Однак, бувають ситуації, коли зникає Інтернет-з’єднання і в цей час дані можуть зникнути. Окрім цього, Telegraf-агент не вміє відсилати дані до різних таблиць, усі ці дані відправлялись би одним потоком без їх групування. Тому, для вирішення цих завдань необхідно використати ПЗ, яка б вміла контролювати потоками з даними і до того ж, могла б гарантувати доставку даних до отримувача. Для цього, обираємо RabbitMQ. RabbitMQ забезпечує гарантовану доставку даних до заданого місця призначення, використовуючи протокол AMQP (Advanced Message Queuing Protocol) [16]. AMQP – це протокол передачі та обробки повідомлень між системами [17]. Головний конкурент для RabbitMQ – Kafka. Однак його конкурент, не може гарантувати

доставку усіх метрик. Для надсилання даних до RabbitMQ з агенту, необхідно буде налаштувати Telegraf на відправку метрик за допомогою AMQP протоколу.

Тепер дані з черг RabbitMQ необхідно відправити до бази даних на зберігання. Для даних, які прив’язані до часу, використання реляційних чи не реляційних баз даних не є гарним вибором, так як при запиту до таблиці при необхідності візуалізації даних витратиться велика кількість часу. Для вирішення цієї проблеми, розробники Telegraf, які працюють у InfluxData, створили програмне забезпечення InfluxDB, який вміє зберігати та робити запити до даних, які були зібрані Telegraf-агентом. InfluxDB - це механізм збереження даних, які прив’язані до часу, забезпечуючи надійність та високу продуктивність [18]. Тому дане програмне забезпечення є найкращим рішенням для різних дій над отриманими метриками.

Заради того, щоб побачити отримані дані з досліджуваного веб-серверу, необхідно скористатися ПЗ, яке б дозволило зображувати дані у вигляді графіків і вміла б робити запити до бази даних, яка розміщена у InfluxDB. Єдиним рішенням є використання графічного інтерфейсу Grafana. Grafana – це програмне забезпечення, яка робить запити до бази даних та відображати у графічному вигляді отримані дані [19]. З її допомогою, ми можемо побачити дані не тільки у текстовому виді, але й у видів графіків, гістерезису, тощо. До того ж, ми можемо легко налаштовувати власні вікна з графіками, як нам буде зручно. Grafana має підтримку використання шаблонів для швидкого їх встановлення у графічний інтерфейс.

3.4 Вибір та обгрунтування застосування апаратних засобів

Перед побудовою серверу моніторингу необхідно обрати компоненти для побудови серверу моніторингу, які б забезпечили довготривалу та надійну роботу. Вони повинні бути сучасними та мати достатньо потужності для того,

щоб робити моніторинг декількох серверів. Так як, постійне оновлення апаратного забезпечення є приводом для втрати важливих даних про стан досліджуваних серверів. Тому, згідно з вимог до забезпечення, які були вказані в минулому розділі для забезпечення безперебійної та швидкої роботи серверу моніторингу, були обрані наступні компоненти.

У якості процесора було обрано – 8-ядерний процесор Intel Core i7-12700K. Його перевагами від 8-ядерного процесора AMD Ryzen 7 5800x є: підтримка оперативної пам’яті нового типу DDR5, наявність 20 потоків проти 16, які будуть корисними при виконанні декількох одночасних операцій, а також збільшений об’єм кеш-пам’яті третього рівня, який становить 25 МБ. До того ж, процесор має вбудовану графіку Intel UHD Graphics 770, що дозволить позбутися потреби купляти відеокарту. До того ж, ця модель процесору від AMD не має вбудовану графіку. Intel Core i7-12700K підтримує технологію віртуалізації Intel Virtualization Technology, яка є необхідною для побудови Docker-контейнерів.

У якості материнської плати було обрано – Gigabyte Z690 Aorus Master. Головними перевагами використання даного компонента: підтримка процесорів Intel під Socket 1700, наявність нового чипсету Intel Z690, а також підтримка оперативної пам’яті типу DDR5. Завдяки їм, забезпечується висока працездатність серверу моніторингу. До того ж, на відміну від плат конкурентів, таких як, Asus та MSI, ця модель має чотири радіатора, що забезпечує відвід тепла. Окрім цього, ця материнська плата має вбудований Wi-Fi адаптер, який можна використати у якості резервного способу для підключення до мережі. Зовнішній вигляд Gigabyte Z690 Aorus Master зображено на рисунку 3.2.

Рисунок 3.2 − Материнська плата Gigabyte Z690 Aorus Master

Так як багатоядерні процесори виділяє велику кількість тепла, необхідно знайти такий кулер, який міг би вирішити цю проблему. Тому, у якості охолодження для процесору було обрано кулер Noctua NH-D15 chromax.black. Даний компонент зарекомендував себе, як найкраще рішення для багатоядерних процесорів, завдяки використанню шести теплових трубок та використанню двох вентиляторів з діаметром 150 мм, що забезпечують потужний повітряний потік. Конкуруючі кулери від be quiet! Dark Pro 4 та Scythe Mugen 5, не мають таких потужний вентиляторів, як на цій моделі від Noctua. Зовнішній вигляд Noctua NH-D15 chromax.black зображено на рисунку 3.3.

Рисунок 3.3 − Кулер Noctua NH-D15 chromax.black

Оперативною пам’яттю для системи контролю завантаженості веб-серверу було обрано Crucial CT16G48C40U5. Дана ОЗП має тип DDR5, що забезпечує високу швидкість системи. До того ж, ця пам’ять має об’єм 16 ГБ, як раз задовольняючи умови видів забезпечення. Дана ОЗП була протестована на взаємодію з материнською платою Gigabyte Z690 Aorus Master її виробниками. Зовнішній вигляд Crucial CT16G48C40U5 зображено на рисунку 3.4.

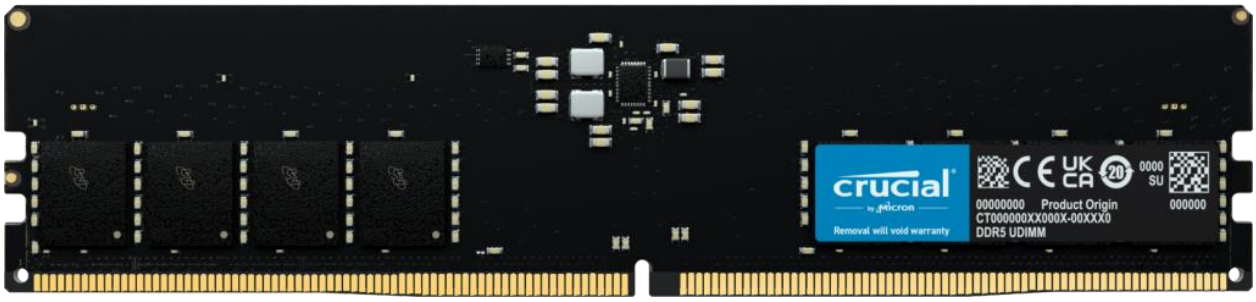

Рисунок 3.4 − Оперативна пам’ять Crucial CT16G48C40U5

У якості накопичувача було обрано Samsung 980 Pro 1 TB. Даний SSD має суттєву перевагу – швидкодія. Так, згідно наданим характеристикам цей накопичувач може читати дані зі швидкістю до 7000 МБ/с, а також записувати дані зі швидкістю до 5000 МБ/с. Це досягається за допомогою підключення Samsung 980 Pro 1 TB до роз’єму M.2, який має доступ до ліній PCI-E 4.0, замість підключення до SATA-портів. Його конкуренти, такі як Western Digital Black SN 850 та Gigabyte Aorus Gen4 мають менші заявлені швидкості. До того ж, цей накопичувач має об’єм 1 ТБ, що задовольняє умови видів забезпечення. Зовнішній вигляд Samsung 980 Pro 1 TB зображено на рисунку 3.5.

Рисунок 3.5 – Накопичувач Samsung 980 Pro 1 TB

Для стабільного функціонування системи контролю завантаженості веб-серверу було обрано блок живлення be quiet! Straight Power 11 Platinum 650W. Потужності цього пристрою достатньо для живлення серверу моніторингу. До того ж, даний блок живлення має усі необхідні роз’єми для підключення інших компонентів серверу. Також, коефіцієнт корисної дії цього компонента становить 99%, що є з одним найкращих показників серед більшості існуючих моделей блоків живлення. Зовнішній вигляд be quiet! Straight Power 11 Platinum 650W зображено на рисунку 3.6.

Рисунок 3.6 – Блок живлення be quiet! Straight Power 11 Platinum 650W

У якості корпусу для сервера моніторингу було обрано Fractal Design Meshify 2 XL Black TG. Даний корпус надає достатньо місця для розташування інших компонентів серверу. Окрім цього, завдяки трьом вбудованим вентиляторам з діаметром 140 мм забезпечується надійна циркуляція повітря. Для ще кращого циркуляції повітря є можливість додавання додаткових вентиляторів. Зовнішній вигляд Fractal Design Meshify 2 XL Black TG зображено на рисунку 3.7.

Рисунок 3.7 – Корпус Fractal Design Meshify 2 XL Black TG

У таблиці 3.1 наведені апаратні характеристики, які буде мати сервер моніторингу.

Таблиця 3.1 – Апаратні характеристики серверу моніторингу

Модель процесора	Intel Core i7-12700K 3.4GHz
Відеокарта	Intel UHD Graphics 770
Материнська плата	Gigabyte Z690 Aorus Master
Охолодження процесора	Noctua NH-D15 chromax.black
Оперативна пам’ять	Crucial CT16G48C40U5 16 GB DDR5-4800
Накопичувач	Samsung 980 Pro 1 TB
Блок живлення	be quiet! Straight Power 11 Platinum 650W
Корпус	Fractal Design Meshify 2 XL Black TG

Наступним компонентом для створення системи контролю завантаженості веб-серверу є маршрутизатор, який би забезпечував зв’язок між сервером моніторингу, веб-сервером та трьома керованими комутаторами. До того ж, роутер повинен мати інтерфейси типу GigabitEthernet, так як пропускна здатність мережі становить 1 Гбіт/сек. Маршрутизатор повинен забезпечувати постійну роботу чотирьох підмереж, при цьому використовуючи протоколи для захисту передачі даних. Підприємство-замовник використовує мережеве обладнання фірми Mikrotik. Тому, згідно існуючих варіантів на ринку для вирішення цієї задачі було обрано маршрутизатор MikroTik RB3011UiAS-RM. Головними перевагами цього маршрутизатору є: наявність 10 портів GigabitEthernet; підтримка протоколів DNS та NAT; наявність протоколу захисту передачі даних IPSec, а також підтримка протоколу L2TP, який забезпечує безпеку передачі даних у віртуальних локальних мережах. У таблиці 3.2 наведені апаратні характеристики роутеру MikroTik RB3011UiAS-RM.

Таблиця 3.2 – Технічні характеристики MikroTik RB3011UiAS-RM

Інтерфейси	10 x GigabitEthernet, 1 x SFP, 1 x RJ-45, 1x USB 3.0 тип A
Встановлена ОС	Router OS 5
Кількість ядр процесору	2
Частота процесору	1.4 ГГц
Підтримка протоколів	DHCP, NAT, IPSec, L2TP, PPPoE, PPTP

Після цього, необхідно підібрати комутатор для забезпечення роботи ізольованих віртуальних мереж між трьома командами розробників IT-компанії. Кожна команда має 23 працівника, тому для забезпечення роботи найманців, майбутній комутатор повинен мати не менш чим 23 інтерфейси типу FastEthernet або вище. Для створення ізольованих підмереж використовують технологію VLAN. Для створення віртуальних мереж повинні використовуватися керовані комутатори, так як вони мають підтримку технології VLAN. Як вже було зазначено вище, підприємство-замовник використовує мережеве обладнання фірми Mikrotik. Таким чином, для вирішення цього завдання був обраний керований комутатор MikroTik GSS326-24G-2S+RM. Головними перевагами цього пристрою є: наявність 24 портів GigabitEthernet; можливість створення віртуальних мереж; фільтрація MAC-адресів; переадресація портів. У таблиці 3.3 наведені технічні характеристики комутатору MikroTik GSS326-24G-2S+RM.

Таблиця 3.3 – Технічні характеристики MikroTik GSS326-24G-2S+RM

Інтерфейси	24x 10/100/1000 Ethernet, 2x SFP
Встановлена ОС	SwOS
Об’єм накопичувача, МБ	2

Продовження таблиці 3.3

Тип накопичувача	Flash
Підтримка протоколів	RSTP, SNMP
Додаткові можливості	фільтрація MAC, підтримка VLAN, листи контролю доступу

3.5 Синтез структурної схеми системи за заданими показниками

На основі попередніх пунктів цього розділу було побудовано результуючу структурну схему системи контролю завантаженості веб-серверу, яка зображена на рисунку 3.8.

У даній структурній схемі на рисунку 3.8. зображено зв’язок компонентів системи контролю завантаженості веб-серверу, які знаходяться у серверній кімнаті. До центрального маршрутизатора з’єднані: три комутатори, веб-сервер, сервер моніторингу та під’єднаний провайдер. Провайдер забезпечує доступ всього офісу до глобальної мережі. Використовуючи центральний маршрутизатор сервер моніторингу та веб-сервер можуть обмінюватися даними та запитами між собою. Три комутатори, які під’єднані до маршрутизатору, забезпечують утворення трьох віртуальних мереж для трьох команд, що забезпечують надійність та безпеку передачі даних.

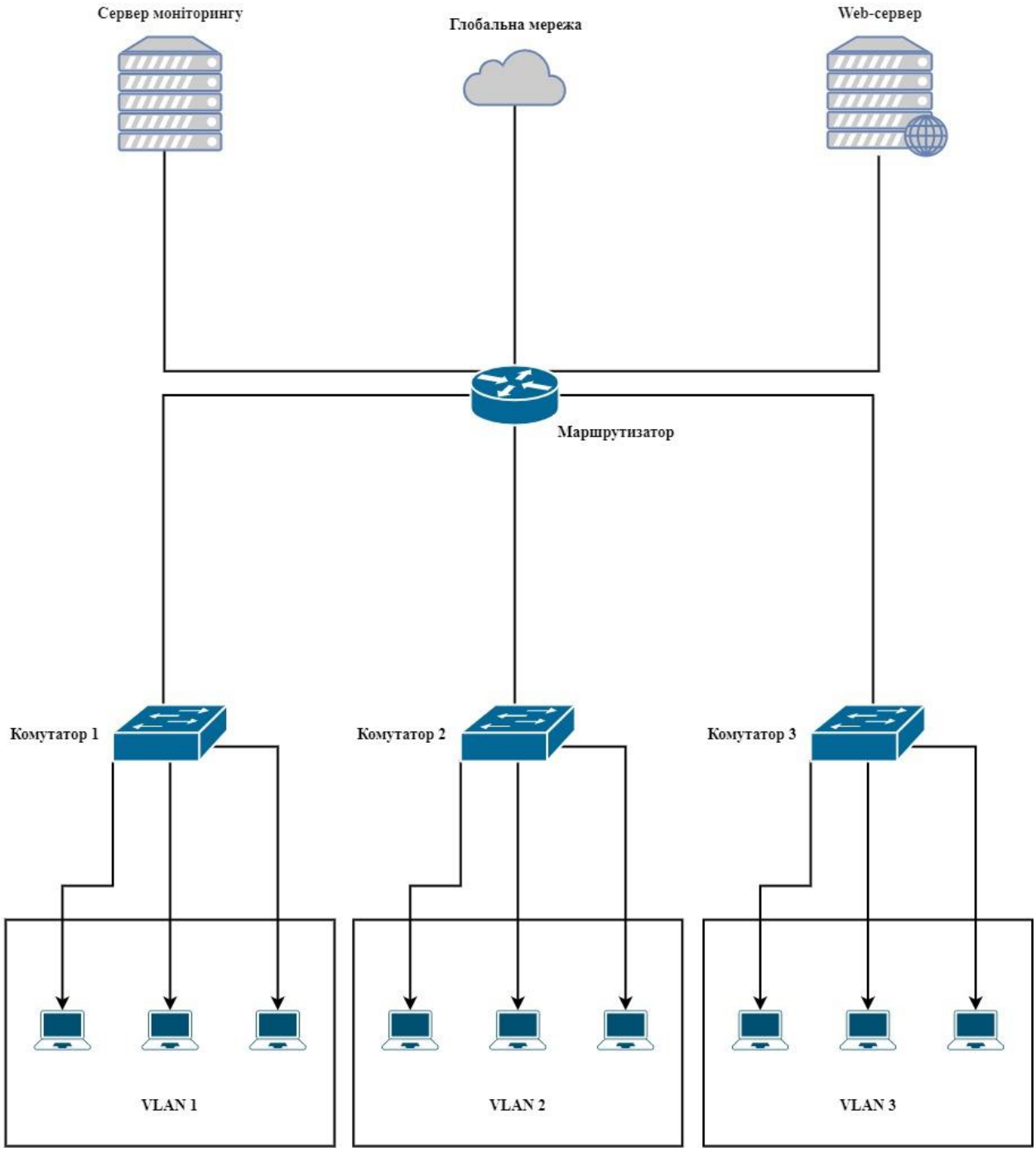

Рисунок 3.8 – Результуюча структурна схема системи контролю завантаженості веб-серверу

3.6 Висновки до розділу

В даному розділі були розроблені вимоги до реалізації системи, безпеки та функціоналу контролю завантаженості веб-серверу, які необхідні для надійної та стабільної роботи. Були розроблені схема функціональної

структури, яка необхідна при побудові системи моніторингу та структурна схема, яка зображує зв’язок між сервером моніторингу та веб-сервером. На основі заданих вимог до видів забезпечення були обрані та встановлені комп’ютерні комплектуючі до серверу моніторингу.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ КОНТРОЛЮ ЗАВАНТАЖЕНОСТІ ВЕБ-СЕРВЕРУ

4.1 Призначення й область застосування програмного забезпечення

Програмне забезпечення системи контролю завантаженості веб-серверу повинно збирати дані про використання досліджуваного серверу. До цих даних відноситься завантаженість наступних компонентів: процесору, оперативної пам’яті та накопичувача.

Дане ПЗ має бути встановлено на сервері, який досліджується та на іншому сервері, який буде зберігати та оброблювати отримані дані. При цьому, забезпечуючи безперервне мережеве з’єднання між ними.

Програмна частина, яка встановлюється на веб-сервері повинна збирати дані за запитом програми, яка встановлена на сервері моніторингу.

Програмна частина, яка встановлюється на сервері моніторингу повинна робити запити до програми, яка працює на досліджуваному сервері за певні проміжки часу, про отримання даних завантаженості заданих компонентів. Після отримання цих метрик, програма має обробити їх та забезпечити зберігання. До того ж, збережені дані мають бути зображені у тому вигляді, який забажає користувач.

Область застосування – IT-компанія Civenty.

4.2 Обґрунтування технічних характеристик програм

Програмне забезпечення системи контролю завантаженості веб-серверу було вирішено розділити на три різні програми, через те, що кожна з них виконує різні завдання.

Перше ПЗ має бути використаною для налаштування нового серверу для зменшення витраченого часу за рахунок зменшення рутинної праці. Вона

повинна встановлювати, видаляти та оновлювати пакети. До того ж, програма має проводити налаштування безпеки та керування запущеними процесами. Окрім цього, до її функцій має ще відноситися керування користувачами. Програма повинна підготувати оточення для розгортання інструментів та файлової структури, які будуть використанні для створення системи контролю завантаженості веб-серверу.

Друге ПЗ буде використано для встановлення на веб-сервер для збору та надсилання даних до серверу моніторингу. Програма повинна видаляти використані Docker-контейнери, які містять налаштування для збору даних з веб-серверу та будувати нові. Агент, який працює усередині контейнера, повинен збирати метрики про завантаженість процесору, оперативної пам’яті та накопичувача досліджуваного серверу. При цьому, забезпечуючи зв’язок між веб-сервером та Docker-контейнером з активним агентом. Отримані дані повинні бути відправлені до серверу моніторингу.

Третє ПЗ буде створено для отримання, обробки, зберігання та візуалізації отриманих метрик з веб-серверу. Програма повинна мати функціонал для отримання, обробки, зберігання та візуалізації отриманих даних з веб-серверу. Для вирішення цієї задачі мають бути застосовані Docker-контейнери для оптимізації використаних ресурсів сервером моніторингу. Тому, програма повинна мати можливості для організації зв’язку контейнерів між собою та фізичним вузлом використовуючи інструмент Docker-Compose. До цього ж, забезпечувати постійну роботу Docker-контейнерів.

Тому, у якості технічного засобу для розробки програми є комп’ютер під керуванням операційної системи CentOS 8, який має встановлену оболонку BASH (Born Again Shell). У якості програмного засобу використовується мова програмування bash та YAML-синтаксис.

4.3 Опис розробленої програми

4.3.1 Загальні відомості

Програма для налаштування серверу моніторингу буде запущеною за допомогою пакету Ansible. Ansible – це програмне забезпечення, яка написана на мові програмування Python, для автоматизації більшості задач у сфері IT, такі як: керування налаштуваннями, встановлення програмних додатків та пакетів, віддаленого керування серверами та іншими [20]. Для Ansible ця програма є плейбуком. Playbook – це набір кроків, які необхідно виконати для досягнення кінцевого стану [21]. Кожен playbook складається з play. Play – це набір декількох task, які мають спільну мету. Task – це дія, яка має бути виконаною. Кожен task використовує один модуль в залежності від задачі. Кожен модуль відповідає за свій тип завдань, наприклад: копіювання файлів, встановлення або видалення пакетів з операційної системи, додавання користувачів, зміна вмісту файлів та багато іншого. У наявності Ansible декілька сотень встановлених модулів, які можуть виконати майже будь-які задачі. Однак, якщо функціоналу існуючих модулів не є достатньою, Ansible надає можливість створити власний модуль. У нашому випадку немає необхідності у створенні нового модулю. Для написання playbook використовується YAML синтаксис.

Друга програма збирає дані про завантаженість веб-серверу. Вона має бути встановлена на досліджуваному сервері. До цієї програми імпортується файл налаштувань в якому вказано, що саме необхідно збирати та на який вузол, треба це відправити. Для написання цього ПЗ було використано мову програмування bash.

Третя програма робить запити до другої програми задля отримання метрик, про завантаженість веб-серверу. Вона має бути встановленою на сервері моніторингу. Після отримання метрик, вона їх оброблює, зберігає та візуалізує. Кожна з цих завдань виконується завдяки Docker-контейнерам. Кожен з цих Docker-контейнерів, має файл налаштувань, який дозволяє підлаштовувати їх функціонал під конкретні цілі. Ця програма описана з

використанням YAML-синтаксису для керування контейнерів та забезпечення між ними зв’язку.

Доступ до редагування цих програм має бути лише у головного системного адміністратора.

4.3.2 Функціональне призначення

Програмне забезпечення системи контролю завантаженості веб-серверу збирає дані про використання системних ресурсів та надсилає їх до серверу моніторингу. Він у свою чергу, забирає метрики з агенту, оброблює їх та зберігає у базі даних. Ця програма надає можливість для візуалізації даних з БД у будь-якому вигляді, який налаштує системний адміністратор.

4.3.3 Опис логічної структури програми

Схема алгоритму програми для налаштування серверу моніторингу зображено на рисунках 4.1, 4.2, 4.3.

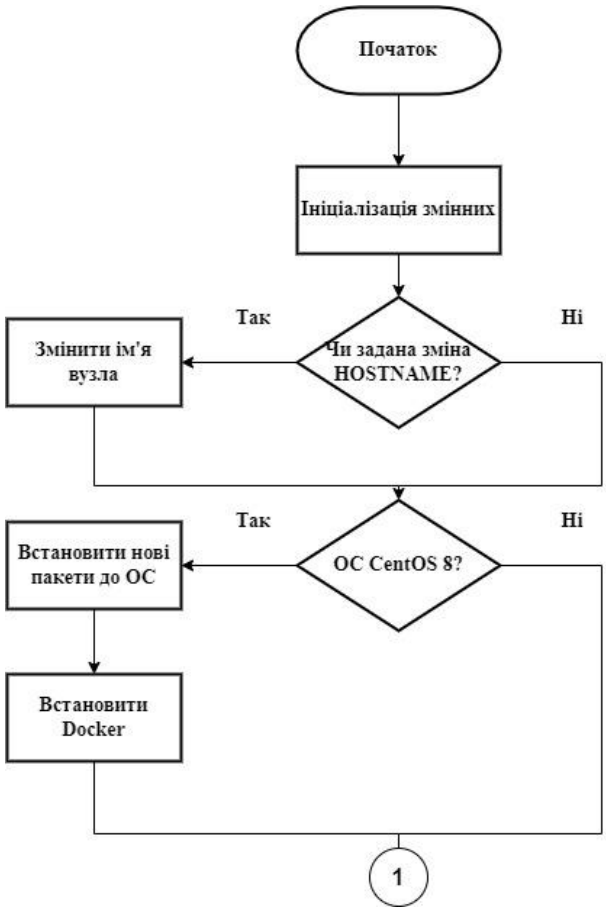

Рисунок 4.1 – Перша частина схеми функціонування програми для налаштування серверу

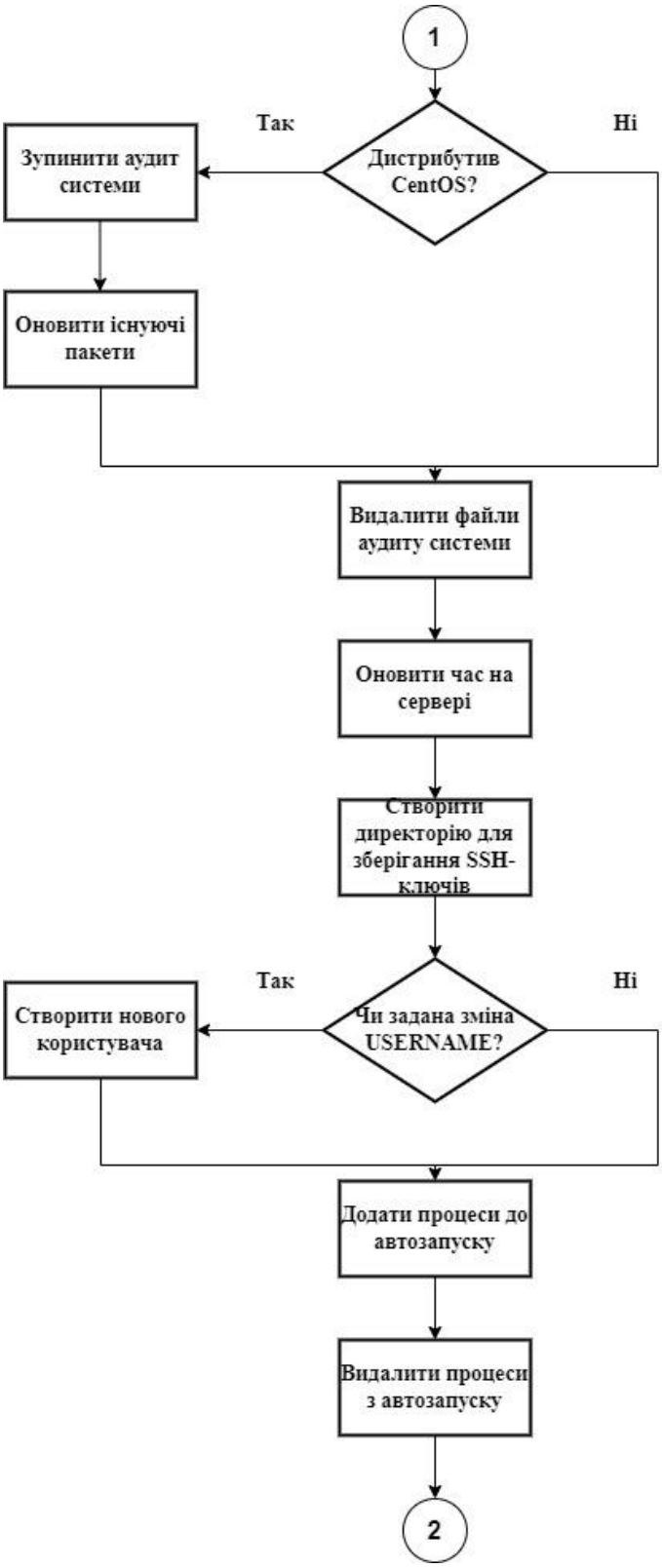

Рисунок 4.2 – Друга частина схеми функціонування програми для налаштування серверу

Рисунок 4.3 – Третя частина схеми функціонування програми для налаштування серверу

Згідно схеми алгоритму, при початку програми вона ініціалізує змінні, які необхідні для виконання майбутніх дій. Після цього, перевіряється чи задана змінна HOSTNAME. Якщо так, то задається нове ім’я для серверу, а якщо ні, то цей крок пропускається. Далі перевіряється чи ця програма запущена на сервері під керуванням операційної системи CentOS 8. Якщо це правда, то встановлюються усі задані пакети та встановлюється Docker, у іншому випадку цей крок не виконується. Якщо на сервері встановлено дистрибутив CentOS, то необхідно зупинити аудит системи, який надмірно споживає системні ресурси та оновити існуючі пакети. Після цього, видаляються накопичені файли аудиту, оновлюється поточний час на сервері та створюється директорія для зберігання SSH-ключів. Якщо у програмі задана зміна USERNAME, то створюється користувач з вмістом цієї змінної. Далі, ПЗ додає необхідні процеси для автозапуску, щоб система контролю завантаженості веб-серверу була справною. Для економії ресурсів серверу, програма видаляє процеси з автозапуску, які не будуть використовуватися.

Після цього, створюється файлова структура, де будуть розміщені необхідні файли для функціонування серверу моніторингу. Після цього кроку встановлюється й Docker-Compose.

Завантаження програми має наступний вигляд:

ansible-playbook ansible-config.yml

Схема алгоритму програми для збору метрик зображено на рисунку 4.4.

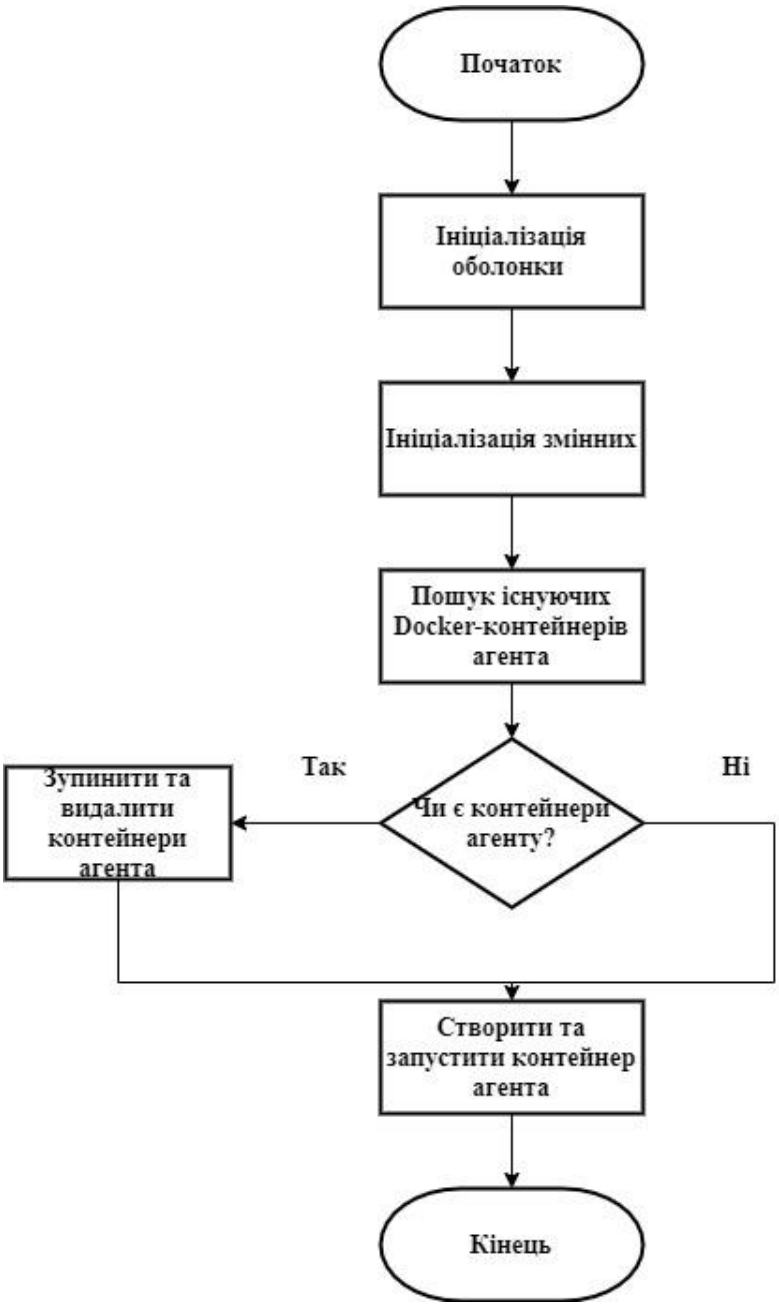

Рисунок 4.4 – Схема функціонування програми збору метрик

Згідно схеми алгоритму, при запуску програми обирається оболонка для її виконання. Після цього, ініціалізуються задані змінні. Далі, програма шукає чи існують вже активні Docker-контейнери, які містять агента для збору метрик з веб-серверу. Якщо цей контейнер існує, то його зупиняють та видаляють. Після цього, створюється та запускається новий Docker-контейнер з агентом для збору метрик з веб-серверу. До контейнеру з агентом імпортується файл налаштувань /home/admin/agent/telegraf.conf, у вмісті якого прописані дані, які необхідно збирати та до якого вузла їх треба відправити.

Схема алгоритму програми для опрацювання метрик зображено на рисунку 4.5.

Згідно схеми алгоритму, при запуску програми обирається версія Docker-Compose для надійної роботи Docker-контейнерів. Після цього, на основі заданих параметрів створюється контейнер для візуалізації даних про завантаженість веб-серверу. До цього Docker-контейнеру імпортуються наступні директорії:

— /home/admin/monitoring/grafana/lib – до нього будуть входити файли, які необхідні для зберігання варіантів шаблонів візуалізації метрик. Тому, у цій директорії буде міститися файл template.json;

— /home/admin/monitoring/grafana/log – у ній будуть зберігатися файли з повідомленнями про помилки;

— /home/admin/monitoring/grafana/plugins – у ній будуть зберігатися файли для встановлення додаткових плагінів. У нашому випадку, існуючого функціоналу достатньо, тому немає необхідності у їх завантаженні.

Далі, на основі заданих вимог для збереження даних створюється Docker-контейнер для збереження отриманих метрик. До цього контейнеру імпортуються імпортуються наступні директорії:

— /home/admin/monitoring/influxdb2/data – у ній будуть зберігатися метрики з баз даних;

— /home/admin/monitoring/influxdb2/config – у ній зберігається файл для налаштування бази даних influxdb2 з назвою influx-configs.

Наступною дією є створення контейнеру, який би забезпечив гарантійну доставку метрик з веб-серверу. Це є корисним, коли з’являються мережеві проблеми і отримані дані з веб-серверу будуть втрачені. До цього контейнеру імпортуються наступні директорії:

— /home/admin/monitoring/rabbitmq/etc – у ній будуть зберігатися файли для налаштувань rabbitmq, до них відносяться такі файли: rabbitmq.conf, enabled_plugins, definitions.json. Файл rabbitmq.conf має опції для налаштування клієнту. Файл definitions.json має налаштування для автоматичного створення черг для надсилання метрик при завантаженні контейнеру. Це економить купу часу, так як до цього, при перезапуску контейнеру для створення черг необхідно було підключатися до графічного клієнту у браузері та створювати усе спочатку. Файл enabled_plugins має список підключених плагінів, які необхідні для функціонування контейнеру з rabbitmq.

Далі, створюється Docker-контейнер, який буде підтримувати зв’язок з досліджуваним веб-сервером, у той же час виконуючи запити до агенту та отримувати від нього метрики. До цього контейнеру імпортується файл налаштувань /home/admin/monitoring/telegraf.conf, у вмісті якого прописані посилання для запиту агента, який встановлений на веб-сервері та до яких черг треба надіслати отримані дані.

Після створення контейнерів необхідним кроком є створення приватної мережі між ними для забезпечення зв’язку, щоб передавалися отримані дані про завантаженість веб-серверу між Docker-контейнерами.

Рисунок 4.5 – Схема функціонування програми опрацювання метрик

Завантаження програми має наступний вигляд:

docker-compose -f docker-compose-server.yml

Опція -f вказує на назву файлу для створення Docker-контейнерів. Головна сторінка інтерфейсу програми зображена на рисунку 4.6.

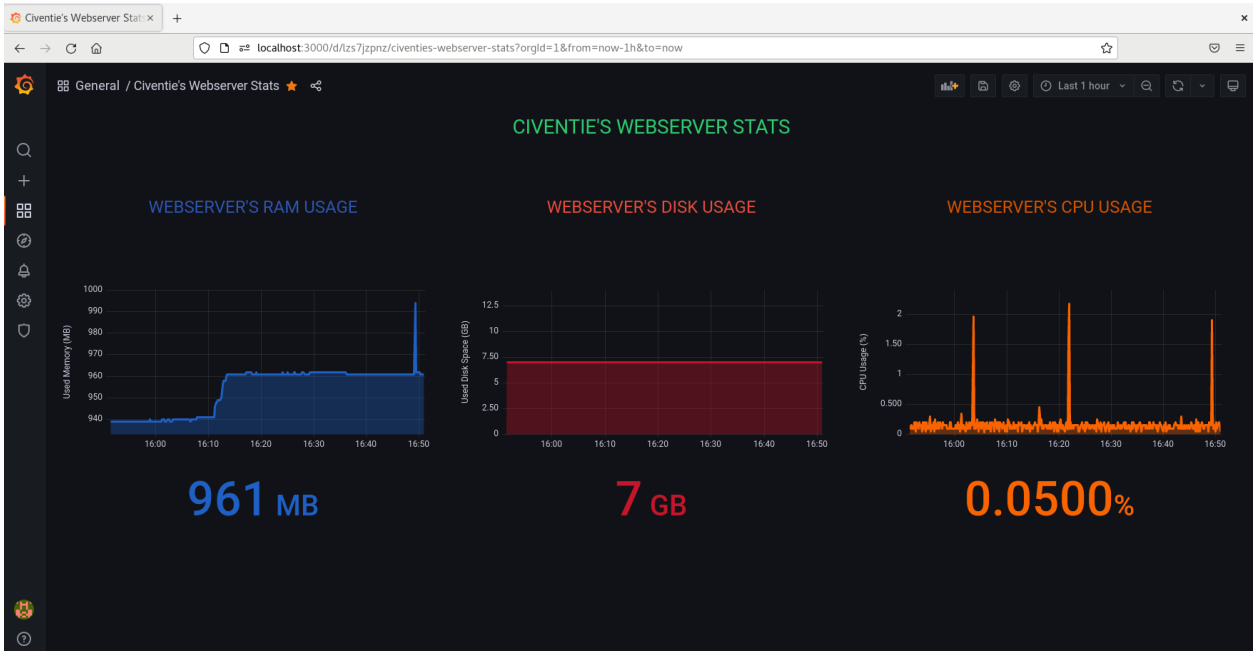

Рисунок 4.6 – Головна сторінка інтерфейсу програми

4.3.4 Опис змінних програм

Програма ansible-playbook.yml має наступні змінні:

— TIMEZONE – використовується для задання часового поясу серверу моніторингу;

— USERNAME – містить у собі ім’я користувача, який буде створений на досліджуваному сервері з привілейованими правами для адміністрування серверу;

— HOSTNAME – використовується для задання імені серверу, який налаштовується.

Програма docker-compose-server.yml має наступні змінні:

— GRAFANA_PORT – зберігає номер TCP-порту, за допомогою якого, буде отримано доступ до графічного інтерфейсу системи контролю завантаженості веб-серверу;

— GRAFANA_USER – містить у собі ім’я користувача, який буде використовуються для доступу до графічного інтерфейсу системи контролю завантаженості веб-серверу;

— GRAFANA_PASSWORD – використовується для задання паролю користувачу для входу до графічного інтерфейсу системи контролю завантаженості веб-серверу;

— GRAFANA_PLUGINS_ENABLED – дозволяє або забороняє встановлювати плагіни для розширення існуючого функціоналу графічного інтерфейсу;

— DOCKER_INFLUXDB_INIT_USERNAME – містить ім’я користувача, від імені якого, буде працювати база даних;

— DOCKER_INFLUXDB_INIT_PASSWORD – зберігає пароль користувача від імені якого, буде працювати база даних;

— DOCKER_INFLUXDB_INIT_ORG – використовується для зберігання імені організації, яка володіє базою даних;

— DOCKER_INFLUXDB_INIT_BUCKET – містить ім’я бази даних, яка буде зберігати системні дані;

— RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS – використовується для імпорту додаткових файлів, при запуску системи контролю черг RabbitMQ. У нашому випадку був імпортований файл definitions.json, який заміняє ручне налаштування RabbitMQ на користь файлу налаштувань.

Програма run-agent.sh має зміну agent_name, яка використовується для задання імені Docker-контейнеру на досліджуваному веб-сервері.

4.4 Очікувані техніко-економічні показники

Використання програмного забезпечення системи контролю завантаженості веб-серверу дозволить відмовитися від витрат на придбання ліцензій для моніторингу серверів. Так як, усі компоненти які входять до створеної системи є безкоштовними. В той же час, один із основних конкурентів ПЗ для моніторингу Nagios, коштує від 1995 доларів в залежності від обраного тарифу [22].

4.5 Висновки до розділу

Для вирішення задачі збирання даних про використання досліджуваного серверу було задані призначення та сфери використання та обґрунтовано технічні характеристики. Це є необхідним для створення завдання технічного завдання для програми. Тому, для її виконання були сплановані три програми:

— налаштування серверу моніторингу, використовуючи YAML-синтаксис;

— збору даних з досліджуваного веб-серверу, використовуючи мову програмування bash;

— опрацювання отриманих даних з веб-серверу, використовуючи YAML-синтаксис.

До кожної з цих програм були розроблені схеми алгоритму. Для автоматизації певних завдань були створені шаблони за допомогою текстового формату JSON.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Формулювання завдання та обґрунтування методики

Мета експерименту: виявлення ефективності роботи системи моніторингу. Дослідити використання програмного забезпечення системи контролю завантаженості веб-серверу. Проаналізувати отримані дані про ефективність впровадження системи моніторингу.

Задача експерименту: використовуючи метод безпосередньої оцінки необхідно виявити значення використаних ресурсів системою контролю завантаженості веб-серверу. При рівних заданих умовах, порівняти з конкуруючою системою. Виявити ефективність використання системи контролю завантаженості веб-серверу.

Сутність методу безпосередньої оцінки полягає в тому, що величина, яка вимірюється визначається за допомогою показниками шкали вимірювального приладу [23]. Цей метод надає значення без додаткових обробок чи надлишкових дій, окрім множення отриманих значень на ціну поділки або сталу вимірювального приладу. Головними перевагами використання методу безпосередньої оцінки є висока швидкість вимірювання та відсутність виконання складних обчислень.

Методика проведення експерименту: Експеримент проводиться за допомогою серверу моніторингу, який налаштований за допомогою програми, яка описана в розділі 4 та встановленої програми htop, яка дозволяє вимірювати споживання системних ресурсів, а саме: процесору, оперативної пам’яті, swap-пам’яті та інші. До налаштованого серверу буде встановлено два програмних забезпечення для моніторингу серверів Zabbix, який є найпопулярнішим рішенням, та системи контролю завантаженості веб-серверу. Запуск та проведення даного експерименту буде проводитися на сервері під керуванням операційної системи CentOS 8. На рисунку 5.1

зображено діаграма варіантів використання системи контролю завантаженості веб-серверу.

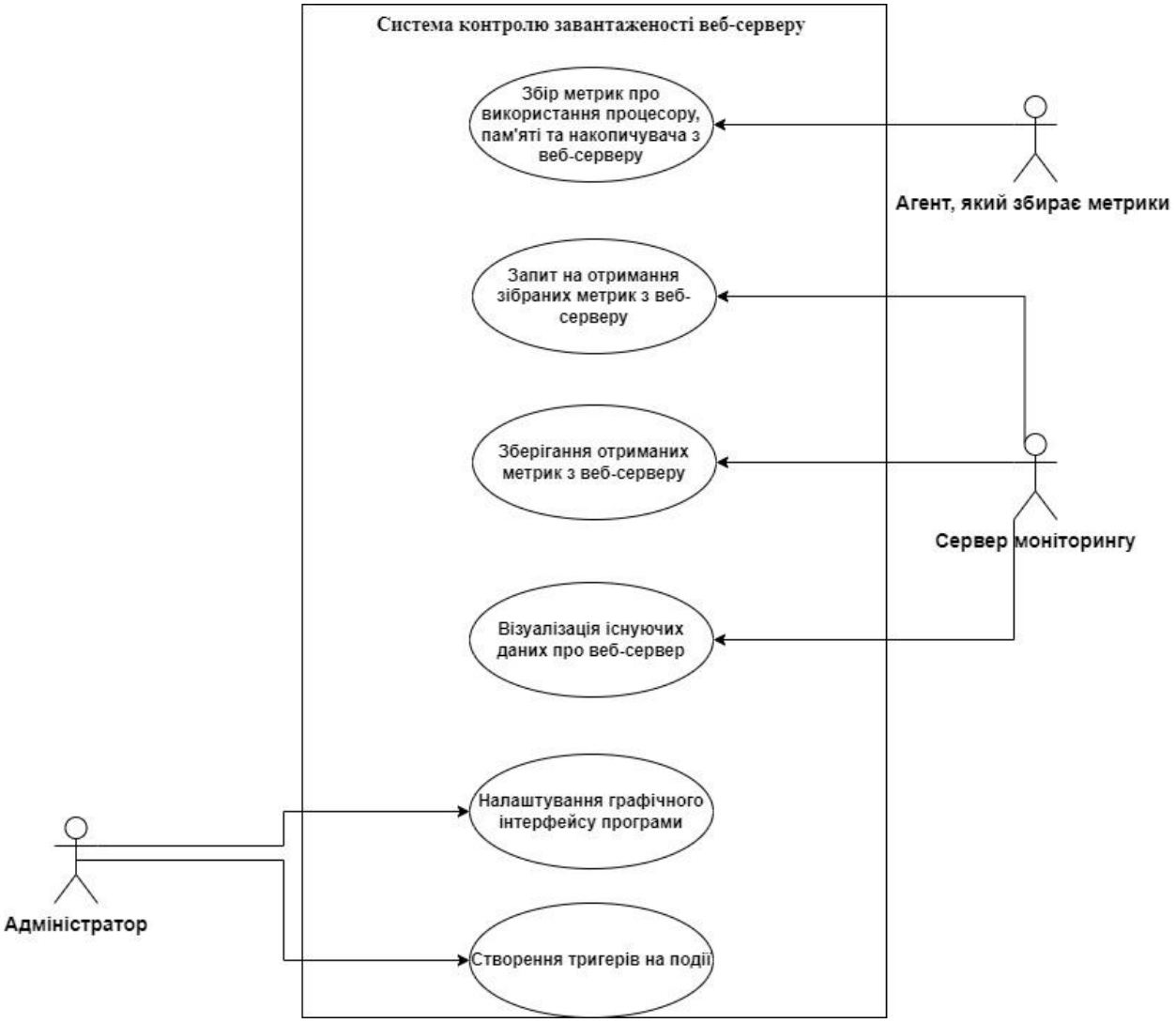

Рисунок 5.1 – Діаграма варіантів використання системи контролю завантаженості веб-серверу

5.2 Вимоги до експерименту

Для проведення експерименту застосовуються наступні вимоги:

— використати систему контролю завантаженості веб-серверу;
— використати ПЗ Zabbix;
— використати сервер зі встановленою ОС CentOS 8;

— використати програму htop для заміру використання системних ресурсів;

— зберегти дані про використання оперативної пам’яті та процесору досліджуваного серверу під час запуску кожної з систем моніторингу;

— розрахувати ефективність використання системи контролю завантаженості веб-серверу.

5.3 Результати експерименту

5.3.1 Сутність експерименту

Даний експеримент виконується за допомогою безпосередньої оцінки, для виявлення максимального значення використання процесору у відсотках та оперативної пам’яті у мегабайтах програмами для контролю завантаженості веб-серверу. Під час експерименту, на протязі 15 хвилин, кожну хвилину буде запускатися програма htop для заміру завантаження процесору та оперативної пам’яті на досліджуваному сервері у реальному часі. Особливу увагу буде примкнуте до максимальних значень використання компонентів серверу. Усвідомлюючи те, що кожна з досліджуваних програм мають той функціонал, який є необхідним для контролю завантаженості веб-серверу ІТ-компанії Civenty. Тому, чим менше буде максимальне число використання процесору та оперативної пам’яті, тим ефективніше працює ПЗ. Після збору отриманих даних про використання системних ресурсів, їх буде занесено до таблиць. На основі даних з таблиць будуть побудовані графіки, які будуть використані для подальшого аналізу.

5.3.2 Результати експерименту у фактах

Використання процесору системою контролю завантаженості веб-серверу наведено у таблиці 5.1.

Таблиця 5.1 – Використання процесору системою контролю завантаженості веб-серверу

№	Час	Завантаженість процесору, %
1	10:01	5.62
2	10:02	5.64
3	10:03	5.61
4	10:04	5.6
5	10:05	5.63
6	10:06	5.63
7	10:07	5.61
8	10:08	5.6
9	10:09	5.61
10	10:10	5.61
11	10:11	5.59
12	10:12	5.62
13	10:13	5.61
14	10:14	5.61
15	10:15	5.59

Використання оперативної пам’яті системою контролю завантаженості веб-серверу наведено у таблиці 5.2.

Таблиця 5.2 – Використання оперативної пам’яті системою контролю завантаженості веб-серверу

№	Час	Використання оперативної пам’яті, МБ
1	10:01	1618.27
2	10:02	1618.08
3	10:03	1623.99

Продовження таблиці 5.2

№	Час	Використання оперативної пам’яті, МБ
4	10:04	1632.04
5	10:05	1625.62
6	10:06	1625.97
7	10:07	1625.14
8	10:08	1625.61
9	10:09	1623.51
10	10:10	1625.32
11	10:11	1625.45
12	10:12	1624.8
13	10:13	1628.29
14	10:14	1627.08
15	10:15	1625.59

Використання процесору системою моніторингу Zabbix наведено у таблиці 5.3.

Таблиця 5.3 – Використання процесору системою моніторингу Zabbix

№	Час	Завантаженість процесору, %
1	10:16	6.14
2	10:17	6.12
3	10:18	6.1
4	10:19	6.09
5	10:20	6.08
6	10:21	6.09
7	10:22	6.09

Продовження таблиці 5.3

№	Час	Завантаженість процесору, %
8	10:23	6.09
9	10:24	6.1
10	10:25	6.08
11	10:26	6.08
12	10:27	6.11
13	10:28	6.1
14	10:29	6.1
15	10:30	6.09

Використання оперативної пам’яті системою моніторингу Zabbix наведено у таблиці 5.4.

Таблиця 5.4 – Використання оперативної пам’яті системою моніторингу Zabbix

№	Час	Використання оперативної пам’яті, МБ
1	10:16	2602.1
2	10:17	2605.59
3	10:18	2602.84
4	10:19	2610.16
5	10:20	2597.78
6	10:21	2607.26
7	10:22	2617.45
8	10:23	2623.29
9	10:24	2619.01
10	10:25	2618.74

Продовження таблиці 5.4

№	Час	Використання оперативної пам’яті, МБ
11	10:26	2622.98
12	10:27	2624.97
13	10:28	2627
14	10:29	2625.01
15	10:30	2626.58

5.3.3 Аналіз відповідності досліджень

У проміжку між 10:00 та 10:15 на досліджуваному сервері було запущено систему контролю завантаженості веб-серверу. У проміжку між 10:15 та 10:30 на досліджуваному сервері було запущено систему моніторингу Zabbix.

Під час виконання кожного з досліджень було щохвилинно записано використання процесору та оперативної пам’яті на сервері, якому був використаний для експериментів. Даний експеримент продемонстрував перевагу у використанні системи контролю завантаженості веб-серверу, а саме у використанні процесору та оперативної пам’яті, при виконанні однакових функцій.

ПЗ, яке досліджує веб-сервер ІТ-компанії Civenty мало максимальне значення використання процесору – 5.64%, натомість система моніторингу Zabbix могла використовувати до 6.14%. Різниця становить 0.5% на користь системи контролю завантаженості власної розробки. Якщо розрахувати середнє споживання процесору за 15 хвилин кожної з програм, то система контролю завантаженості веб-серверу споживала у середньому – 5.61 %, натомість Zabbix – 6.1%. Різниця становить 0.49 %. До того ж, існує ймовірність, що відрив між ПЗ у споживанні процесору може бути збільшений при моніторингу, більш чим 1 серверу. На рисунку 5.2 зображено графік використання процесору програмами для візуального порівняння.

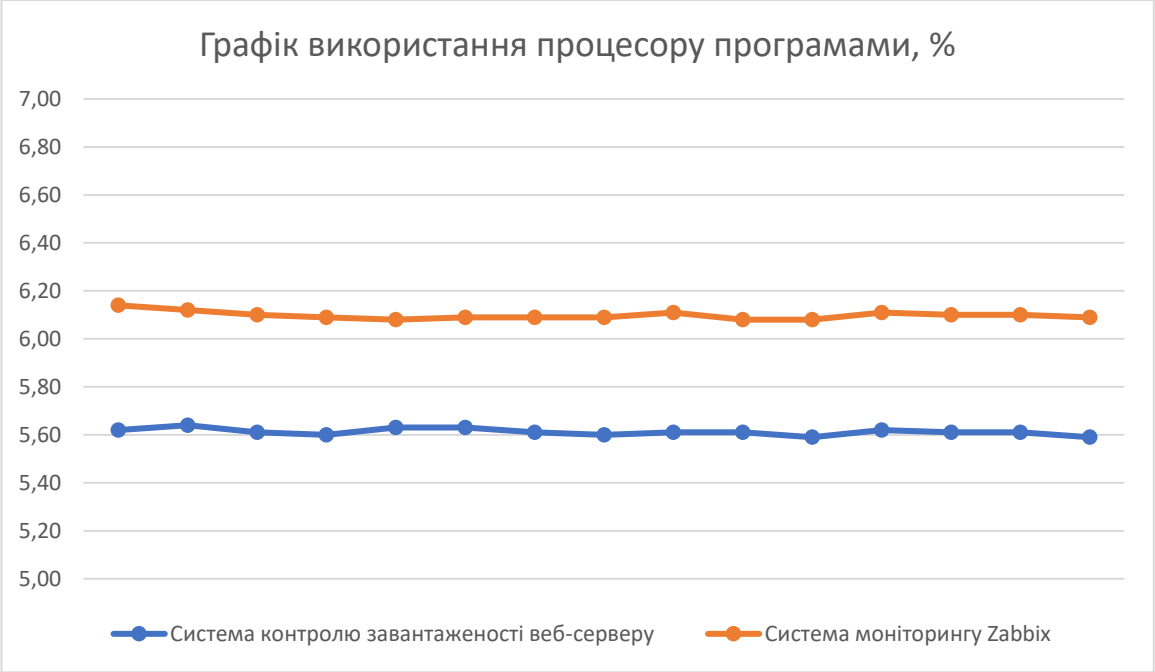

Рисунок 5.2 – Графік використання процесору програмами

Якщо порівнювати значення у графіку використання оперативної пам’яті між цими програмами, то тут різниця набагато більша. ПЗ, яке досліджує веб-сервер ІТ-компанії Civenty мало максимальне значення використання оперативної пам’яті – 1632.04 МБ, натомість система моніторингу Zabbix могла використовувати до 2627 МБ. Різниця становить 994.96 МБ або 62.12% на користь системи контролю завантаженості власної розробки. Якщо розрахувати середнє споживання оперативної пам’яті за 15 хвилин кожної з програм, то система контролю завантаженості веб-серверу споживала у середньому – 1624.98 МБ ОЗП, натомість Zabbix – 2615.38 МБ ОЗП. Різниця становить 990.4 МБ або 62.13%. Тут також, існує ймовірність, що відрив між ПЗ у використанні оперативної пам’яті може бути збільшений при моніторингу, більш чим 1 серверу. На рисунку 5.3 зображено графік використання оперативної пам’яті програмами для візуального порівняння.

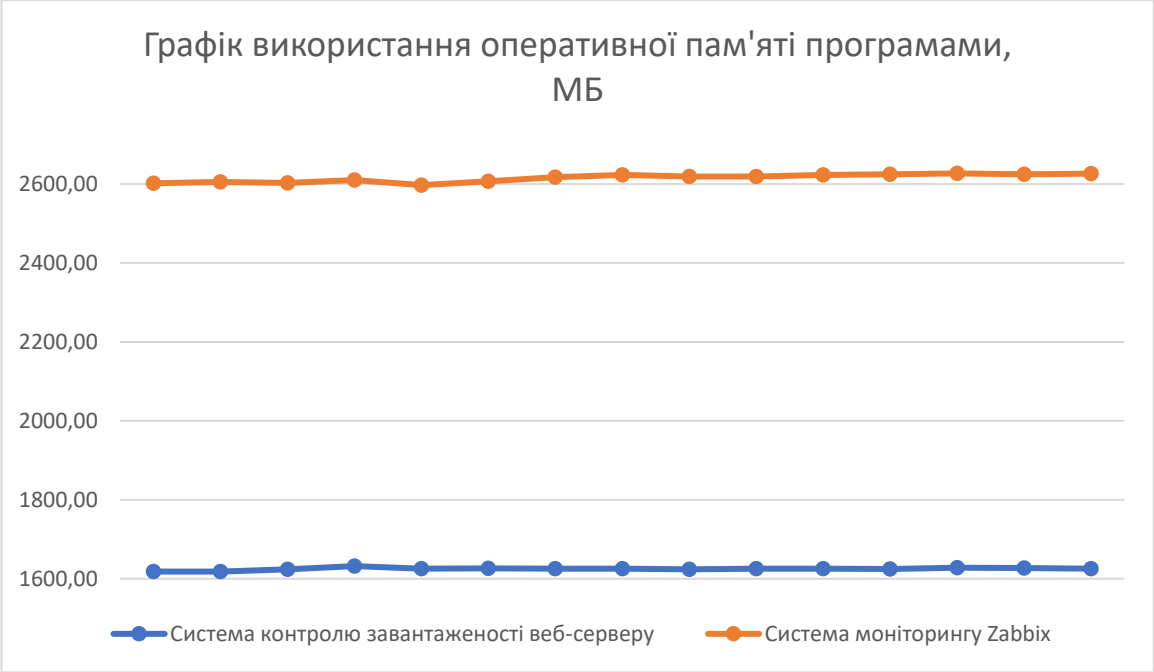

Рисунок 5.3 – Графік використання оперативної пам’яті програмами

5.3.4 Характеристика новизни результатів

Наукове значення експерименту, який проводився з системою контролю завантаженості веб-серверу полягало у обґрунтуванні доцільності її використання. Практичне значення результатів експерименту полягає в тому, що дану програму можливо запровадити, як рішення для моніторингу серверів.

5.4 Висновки до розділу

У даному розділі перед початком експерименту було обрано її мету та задачу. До того ж, було обрано методику проведення експерименту. Перед початком проведення експерименту було важливо задати вимоги до неї. Також було описано процес її виконання. Експеримент пройшов вдало, до того ж, вдалось обґрунтувати використання системи контролю завантаженості серверів власної розробки, поміж аналогів. Дана система продемонструвала достатній функціонал для моніторингу компонентів веб-серверу, при тому,

використовуючи менші системні ресурси, чим конкуренти. Така економія ресурсів дозволить досліджувати більшу кількість серверів.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача створення програмно-технічних засобів комп’ютерної системи контролю завантаженості веб-серверу ІТ-компанії Civenty. Дана система була розроблена за рахунок аналізу існуючих систем моніторингу, розробки моделей з використанням методів аналізу та синтезу, побудови моделей для КС, а також розробки програмного забезпечення. Завдяки правильно заданим вимогам було підібрані та обґрунтовані обладнання для системи контролю завантаженості веб-серверу.

Основні висновки і результати роботи полягають у наступному:

1. Використовуючи метод аналізу та синтезу, а також виконавши аналіз існуючих систем моніторингу були виділені основні переваги, які треба запровадити до власної системи моніторингу, при цьому мінуючи найпоширеніші недоліки, які впливають на працездатність системи;
2. Перед розробкою програмного забезпечення для моніторингу веб-сервера була розроблена математична модель з використанням методу Ланцюг Маркова. З його допомогою, були розраховані стани системи при різних отриманих даних від досліджуваного серверу, які будуть використанні при розробці програм;
3. Для забезпечення стабільної роботи системи контролю завантаженості веб-серверу, були створені та затверджені вимоги до обладнання. На основі цих вимог, були обрані: три комутатори, один маршрутизатор, та компоненти для серверу. І як результат була отримана результуюча структурна схема системи контролю завантаженості веб-серверу, на якій зображено зв’язок між компонентами системи;
4. Розробка програмного забезпечення проводилась у додатку vim, який вміє працювати з майже усіма мовами програмування, за допомогою використання плагінів для розширення функціоналу;

5. За допомогою використання YAML синтаксису були написані програми, які дозволяють автоматично налаштувати та встановити сервер моніторингу. Це дозволяє зменшити витрати часу на рутину роботу, а також забезпечує легке масштабування компонентів системи моніторингу;

6. Для проведення експерименту були задані мета, завдання, а також методику експерименту. Експеримент пройшов вдало, так як, усі задані умови були виконані. До того ж, були отримані дані, які були проаналізовані , що дозволяють обґрунтувати використання власної системи моніторингу;

7. Результатом експерименту є доцільність використання власної системи контролю завантаженості веб-серверу, поміж існуючих конкурентів, завдяки виконанню поставлених задач до системи моніторингу, при цьому споживаючи менше апаратних ресурсів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Основы мониторинга и сбора метрик. URL: https://www.8host.com/blog/osnovy-monitoringa-i-sbora-metrik/ (дата звернення: 04.10.2021);

2. 15 Best IT Infrastructure Monitoring Tools & Software. URL: https://sematext.com/blog/infrastructure-monitoring-tools/#toc-10-nagios-9 (дата звернення: 04.10.2021);

3. Using munin plugins. URL: http://guide.munin-monitoring.org/en/latest/plugin/use.html (дата звернення: 06.10.2021);

4. Nagios vs. Cacti. URL: https://www.osradar.com/nagios-vs-cacti/ (дата звернення: 06.10.2021);

5. ABOUT US. URL: https://civenty.com/about (дата звернення: 09.10.2021);

6. Элементы данных. URL: https://www.zabbix.com/documentation/current/ru/manual/config/items (дата звернення: 09.10.2021);

7. МАТЕМАТИЧНІ МОДЕЛІ: ОЗНАЧЕННЯ, ХАРАКТЕРИСТИКИ, ЕТАПИ ПОБУДОВИ. URL: https://web.posibnyky.vntu.edu.ua/firen/3zlepko_osnovy_biomedychnogo_radioelektronnogo_aparatobuduvannya/6.html (дата звернення: 09.10.2021);

8. Markov chain. URL: https://www.lexico.com/en/definition/markov_chain (дата звернення: 10.10.2021);

9. Simulink. URL: https://www.csoft.ru/catalog/soft/simulink/simulink.html (дата звернення: 10.10.2021);

10. %YAML 1.2. URL: https://yaml.org/ (дата звернення: 01.11.2021);

11. Docker Installation and Configuration Requirements. URL: https://docs.cambridgesemantics.com/anzograph/v2.2/userdoc/install-docker.htm (дата звернення: 01.11.2021);

12. Install Docker Engine. URL: https://docs.docker.com/engine/install/#server (дата звернення: 01.11.2021);

13. Cryptography Today. URL: https://web.archive.org/web/20160305203915/https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml (дата звернення: 01.11.2021);

14. Create a strong password & a more secure account. URL: https://support.google.com/accounts/answer/32040?hl=en#zippy=%2Cmake-your-password-longer-more-memorable (дата звернення: 03.11.2021);

15. Telegraf. URL: https://www.influxdata.com/time-series-platform/telegraf/ (дата звернення: 04.11.2021);

16. RabbitMQ. URL: https://www.rabbitmq.com/ (дата звернення: 07.11.2021);

17. Part 1: RabbitMQ for beginners – What is RabbiMQ? URL: https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html (дата звернення: 07.11.2021);

18. InfluxDB. URL: https://www.influxdata.com/products/influxdb/ (дата звернення: 10.11.2021);

19. Grafana. URL: https://grafana.com/grafana/ (дата звернення: 12.11.2021);

20. Overview How Ansible Works. URL: https://www.ansible.com/overview/how-ansible-works (дата звернення: 22.11.2021);

21. Working With Playbooks. URL: https://docs.ansible.com/ansible/2.9/user_guide/playbooks.html (дата звернення: 22.11.2021);

22. Nagios XI Enterprise Server and Network Monitoring Software. URL: https://www.nagios.com/products/nagios-xi/ (дата звернення: 24.11.2021);

23. Метод безпосередньої оцінки. URL: http://ni.biz.ua/8/8_14/8_140470_metod-neposredstvennoy-otsenki.html (дата звернення: 09.12.2021).

ДОДАТОК А

Текст програми системи контролю завантаженості веб-серверу.

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОНТРОЛЮ ЗАВАНТАЖЕНОСТІ ВЕБ-СЕРВЕРУ

ТЕКСТ ПРОГРАМИ

804.02070743.22007-01.12 01

Листів 16

2022

АНОТАЦІЯ

У цьому документі представлено код програмного забезпечення системи контролю завантаженості веб-серверу.

Програми реалізовані використовуючи YAML-синтаксис та мову програмування bash.

Середовище для розробки програм – текстовий редактор Vi

ЗМІСТ

Ansible-playbook.yml

```
# Описуємо параметри play, який буде виконаний на локальному хості
- hosts: 127.0.0.1
  connection: local
# Дозволити виконання task від імені суперкористувача
become: true
# Ініціалізуємо змінні
vars:
   - TIMEZONE: "Europe/Kiev" # Задаємо часовий пояс для серверу
   - USERNAME: "admin" # Задаємо ім’я для створення нового користувача
   - HOSTNAME: "monitoring" # Задаємо ім’я вузла

# Створюємо масив tasks
 tasks:
# Виконати task зі зміни ім’я хосту на сервері з використанням модулю hostname, якщо задана змінна HOSTNAME
   - name: Change hostname
     hostname:
       name: "{{ vars.HOSTNAME }}"
     when: HOSTNAME is defined

# Виконати блок згрупованих tasks, якщо на сервері встановлена ОС CentOS 8
  - block:
     # Виконати task зі встановленням необхідних пакетів на сервері за допомогою модулю yum
    - name: Install minor utilites for CentOS 8
      yum:
        name:
  - htop
  - iotop
  - atop
  - dstat
  - logwatch
  - smartmontools
  - lsof
  - tcpdump
  - unzip
```

```
  - pciutils
  - mcelog
        - net-tools
        - glances
        - iptables-services
        - irqbalance
        - tmux
        - psmisc
        - chrony
        - bzip2
        - firewalld
        - yum-utils
        - glibc-langpack-ru
      state: latest
    # Завантажити необхідні залежності для встановлення Docker, за допомогою модулю yum
    - name: "Installing Docker Prerequisite packages"
      yum:
        name: "{{ item }}"
        state: latest
      with_items:
        - yum-utils
        - device-mapper-persistent-data
        - lvm2
    - name: "Configuring docker-ce repo"
# Додавання репозиторію для завантаження docker за допомогою модулю get_url
      get_url:
        url: https://download.docker.com/linux/centos/docker-ce.repo
        dest: /etc/yum.repos.d/docker-ce.repo
        mode: 0644
    # Встановлення Docker на сервер останньої версії за допомогою модулю yum
    - name: " Installing Docker latest version"
      yum:
        name: docker-ce
        state: latest
    # Додати користувача admin до групи docker за допомогою модулю user
    - name: Add user admin to docker group
      user:
        name: "{{ vars.HOSTNAME }}"
        groups: docker
        append: yes
    when: ansible_distribution == "CentOS" and ansible_distribution_major_version == "8"
```

```
    # Виконати блок згрупованих tasks, якщо на сервері встановлена ОС CentOS будь-якої версії
    - block:
      # Зупинити процес auditd за допомогою модулю shell
      - name: Stop auditd
        shell: service auditd stop
        args:
          warn: false
      # Видалити зайві пакети за допомогою модулю shell
      - name: Delete audit packages
        yum:
          name:
            - selinux*
            - auditd
          state: absent
      # Редагувати файл налаштувань selinux
      - name: Disable selinux
        replace:
          path: /etc/selinux/config
          regexp: 'SELINUX=[[:alpha:]]\+'
          replace: 'SELINUX=disabled'
    # Оновити усі існуючі пакети на сервері за допомогою модулю yum
      - name: Update all packages
        yum:
          name: '*'
          state: latest
        when: ansible_distribution == "CentOS"

      # Видалити тимчасові файли аудиту з директорій /var/log/audit та /etc/selinux/targeted
      - name: Remove se files
        file:
          path: "{{ item }}"
          state: absent
          force: yes
        with_items:
          - /var/log/audit
          - /etc/selinux/targeted

    # Виконати блок згрупованих tasks для налаштування поточного часу
      - block:
        - name: Create a localtime link
          file:
```

```
      src: /usr/share/zoneinfo/{{ vars.TIMEZONE }}
      dest: /etc/localtime
      state: link
      force: yes
    - name: Update timezone
      shell: echo {{ vars.TIMEZONE }} > /etc/timezone

  # Виконати блок згрупованих tasks для створення директорії з ssh-ключами для всіх користувачів
    - block:
      # Створити директорію для зберігання ssh-ключів користувачу за допомогою модулю file. Доступ до цієї директорії надати тільки до власника цієї директорії
      - name: Create a ssh directory
        file:
          path: /etc/skel/.ssh
          state: directory
          mode: '0700'
      # Створити файл для зберігання публічних ssh-ключів користувачу за допомогою модулю file. Доступ до цього файлу надати тільки до власника цього файлу без можливості запуску, як виконавчий файл
      - name: Create a ssh-key file
        file:
          path: /etc/skel/.ssh/authorized_keys
          state: touch
          mode: '0600

  # Виконати блок згрупованих tasks для створення нового користувача
    - block:
      # Створити нового користувача за допомогою модулю user та додати його до групи wheel
      - name: Add user
        user:
          name: "{{ vars.USERNAME }}"
          group: wheel
      # Додати публічний ssh-ключ у кінець файлу authorized keys для нового користувача, використовуючи модуль lineinfile
      - name: Add a ssh-key to new user
        lineinfile:
          path: /home/{{ vars.USERNAME }}/.ssh/authorized_keys
          create: yes
          insertafter: EOF
          line: key
      when: USERNAME is defined
```

```
# Додати процеси до автозапуску, за допомогою модулю service
  - name: Enable services for automatic start
    service:
      name: "{{ item }}"
      enabled: yes
    with_items:
      - chronyd
      - iptables
      - smartd
      - mcelog
      - atop

# Видалити процеси з автозапуску, за допомогою модулю service
  - name: Disable services for automatic start
    service:
      name: firewalld
      state: stopped
      enabled: no

# Створити файлову структуру системи контролю завантаженості веб-серверу за допомогою модулю file
  - name: Create a project directory
    file:
      path: "{{ item }}"
      state: directory
      mode: '0775'
      owner: {{ vars.USERNAME }}
      group: {{ vars.USERNAME }}
    with_items:
      - "/home/{{ vars.USERNAME }}/monitoring"
      - "/home/{{ vars.USERNAME }}/monitoring/influxdb2"
      - "/home/{{ vars.USERNAME }}/monitoring/influxdb2/data"
      - "/home/{{ vars.USERNAME }}/monitoring/influxdb2/config"
      - "/home/{{ vars.USERNAME }}/monitoring/rabbitmq"
      - "/home/{{ vars.USERNAME }}/monitoring/rabbitmq/etc"
      - "/home/{{ vars.USERNAME }}/monitoring/rabbitmq/data"
      - "/home/{{ vars.USERNAME }}/monitoring/rabbitmq/log"
      - "/home/{{ vars.USERNAME }}/monitoring/telegraf"
      - "/home/{{ vars.USERNAME }}/monitoring/grafana"
      - "/home/{{ vars.USERNAME }}/monitoring/grafana/lib"
      - "/home/{{ vars.USERNAME }}/monitoring/grafana/log"
```

```
      - "/home/{{ vars.USERNAME }}/monitoring/grafana/plugins"


    # Встановлення docker-compose за допомогою модулю get_url
    - name: Install docker-compose
      get_url:
        url: https://github.com/docker/compose/releases/download/1.29.2/docker-compose-{{ ansible_system }}-{{ ansible_userspace_architecture }}
        dest: /usr/local/bin/docker-compose
        mode: 'u+x,g+x'
    when: ansible_distribution == "CentOS"
```

Docker-compose-server.yml

```
version: "3.5"

services:

# Створюємо контейнер з ім’ям grafana

  grafana:

    image: grafana/grafana:latest

    container_name: grafana

    restart: always

    links:

      - influxdb

    ports:

      - "127.0.0.1:3000:3000"

    volumes:

      - /home/admin/monitoring/grafana/lib:/var/lib/grafana

      - /home/admin/monitoring/grafana/log:/var/log/grafana

      - /home/admin/monitoring/grafana/plugins:/var/lib/grafana/plugins

    environment:

      - GRAFANA_PORT=3000 # Задаємо порт для з’єднання до графічного інтерфейсу моніторингу

      - GRAFANA_USER=<value> # Задаємо ім’я користувача від імені якого можливо отримати доступ до графічного інтерфейсу моніторингу
```

- GRAFANA_PASSWORD=<value> # Задаємо пароль користувача для входу користувача до графічного інтерфейсу моніторингу

- GRAFANA_PLUGINS_ENABLED=true # Дозволяємо завантаження плагінв для розширення функціоналу графічного інтерфейсу

networks:

monitoring:

ipv4_address: 192.168.0.5

Створюємо контейнер з ім’ям influxdb

influxdb:

container_name: influxdb2-server

image: influxdb:2.0

restart: always

links:

- telegraf-server

ports:

- "127.0.0.1:8086:8086"

volumes:

- /home/admin/monitoring/influxdb2/data:/var/lib/influxdb2

- /home/admin/monitoring/influxdb2/config:/etc/influxdb2

environment:

- DOCKER_INFLUXDB_INIT_USERNAME=<value> # Створюємо користувача, від імені якого буде працювати база даних

- DOCKER_INFLUXDB_INIT_PASSWORD=<value> # Створюємо пароль для користувача, від імені якого буде працювати база даних

- DOCKER_INFLUXDB_INIT_ORG=<value> # Задаємо ім’я організації, якої належить база даних

- DOCKER_INFLUXDB_INIT_BUCKET=<value> # Створюємо базу даних для зберігання даних про завантаження веб-серверу

networks:

monitoring:

```
        ipv4_address: 192.168.0.2


# Створюємо контейнер з ім’ям telegraf-server
  telegraf-server:
    container_name: telegraf-server
    image: telegraf:1.19
    links:
      - rabbitmq
    restart: always
    volumes:
      - /home/admin/monitoring/telegraf-server/telegraf.conf:/etc/telegraf/telegraf.conf
    networks:
      monitoring:
        ipv4_address: 192.168.0.3


# Створюємо контейнер з ім’ям rabbitmq
  rabbitmq:
    container_name: rabbitmq
    image: rabbitmq:3.8
    ports:
      - “127.0.0.15672:5672”
      - “127.0.0.1:15672:15672”
    volumes:
      - /home/admin/monitoring/rabbitmq/etc/:/etc/rabbitmq/
      - /home/admin/monitoring/rabbitmq/data/:/var/lib/rabbitmq/
      - /home/admin/monitoring/rabbitmq/log/:/var/log/rabbitmq/
    environment:
      - RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS=-rabbitmq_management load_definitions
"/etc/rabbitmq/definitions.json" # Задаємо файли, які необхідно імпортувати при початку роботи контейнера
```

```
    networks:
      monitoring:
        ipv4_address: 192.168.0.4
# Створюємо мережу для Docker-контейнерів на сервері моніторингу
networks:
  monitoring:
    driver: bridge
    ipam:
     config:
       - subnet: 192.168.0.0/24
         gateway: 192.168.0.1
```

Definitions.json

```
{ # Ініціалізація версій
  "rabbit_version": "3.8.25",
  "rabbitmq_version": "3.8.25",
  "product_name": "RabbitMQ",
  "product_version": "3.8.25",
  "users": [
    { # Ініціалізація користувачів
      "name": "admin",
      "password_hash": "avRvrh4m0Ym3vTLfeUd/iwE5b+DZkEpwCWhnIxAtlEo75YFp",
      "hashing_algorithm": "rabbit_password_hashing_sha256",
      "tags": "administrator",
      "limits": {}
    }
  ],
```

```
# Налаштування віртуального вузла
  "vhosts": [
    {
      "name": "/"
    }
  ],
  "permissions": [
    {
      "user": "admin",
      "vhost": "/",
      "configure": ".*",
      "write": ".*",
      "read": ".*"
    }
  ],
  "topic_permissions": [],
  "parameters": [],
  "global_parameters": [
    {
      "name": "internal_cluster_id",
      "value": "rabbitmq-cluster-id-la4C-auzVla3wOwtW8D8FQ"
    }
  ],
  "policies": [],
# Створення черг
  "queues": [
    {
      "name": "mem",
      "vhost": "/",
```

```
    "durable": true,
    "auto_delete": false,
    "arguments": {
      "x-queue-type": "classic"
    }
  },
  {
    "name": "cpu",
    "vhost": "/",
    "durable": true,
    "auto_delete": false,
    "arguments": {
      "x-queue-type": "classic"
    }
  },
  {
    "name": "disk",
    "vhost": "/",
    "durable": true,
    "auto_delete": false,
    "arguments": {
      "x-queue-type": "classic"
    }
  }
],
"exchanges": [
  {
    "name": "telegraf",
    "vhost": "/",
```

```
    "type": "topic",
    "durable": true,
    "auto_delete": false,
    "internal": false,
    "arguments": {}
  }
 ],
 "bindings": [
  {
    "source": "telegraf",
    "vhost": "/",
    "destination": "cpu",
    "destination_type": "queue",
    "routing_key": "cpu",
    "arguments": {}
  },
  {
    "source": "telegraf",
    "vhost": "/",
    "destination": "disk",
    "destination_type": "queue",
    "routing_key": "disk",
    "arguments": {}
  },
  {
    "source": "telegraf",
    "vhost": "/",
    "destination": "mem",
    "destination_type": "queue",
```

```
      "routing_key": "mem",
      "arguments": {}
    }
  ]
}
```

Run-agent.sh

```
#!/bin/bash

agent_name=telegraf-client # Задаємо ім’я контейнеру з агентом

# Перевірка чи є існуючі контейнери агента
docker ps | grep -i $agent_name > /dev/null

if [ $? -eq 0 ];
then
  docker stop $agent_name
  docker rm $agent_name
fi
# Створити Docker-контейнер
docker run -id --name $agent_name -v /home/alex/metrics/telegraf-client/telegraf.conf:/etc/telegraf/telegraf.conf:ro telegraf:1.19
```