

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КРИМСЬКИЙ РУСЛАН ОЛЕГОВИЧ

Допускається до захисту:

завідувач кафедри
інформаційних технологій,
доктор технічних наук, доцент

_____ Т. В. Нескородева

« _____ » _____ 20__ р.

**РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ АПАРАТНОГО
ЗАБЕЗПЕЧЕННЯ НА LINUX СЕРВЕРІ**

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:

Антонов Ю.С., доцент кафедри
інформаційних технологій,
к.ф.-м.н., доцент

Оцінка:

_____/_____/_____

(бали за шкалою ЄКТС/за національною шкалою)

Голова

ЕК:

(підпис)

АНОТАЦІЯ

Кримський Р.О. Розробка системи моніторингу стану апаратного забезпечення на Linux сервері. Спеціальність 122 «Комп'ютерні науки». Освітня програма «Сучасні інформаційні технології та програмування». Донецький національний університет імені Василя Стуса, Вінниця 2022.

У кваліфікаційній роботі проаналізовано та розроблено систему для моніторингу апаратного забезпечення на Linux сервері.

Проведено аналіз сучасних методів моніторингу апаратного забезпечення на Linux сервері та інструментів, які дозволяють вирішити поставлену задачу.

Піднятий власний Linux сервер та підключено систему Web-API для моніторингу стану апаратного забезпечення на Linux сервері.

Ключові слова: Linux сервер, Grafana, Prometheus, Docker, Node-exporter, Alert Manager, Telegram bot, Virtual Box, Debian,

46 с., Рис. 31, Табл. 1, Бібліографія: 44 найменування.

ABSTRACT

Krymskyi R.O. Deploy the hardware security monitoring system on a Linux server. Specialty is 122 «Computer Sciences». Vasil' Stus Donetsk National University, Vinnitsa 2022.

In qualification work the system for their monitoring is analyzed and developed.

An analysis of modern methods of hardware monitoring on Linux servers and tools that allow virtualization of installed calibers. Raised Linux server and connected Web-API system to monitor hardware status on Linux server.

Keywords: Linux server, Grafana, Prometheus, Docker, Node-exporter, Alert Manager, Telegram bot, Virtual Box, Debian.

46 p. Fig. 31, Table. 1, Bibliography.: 44 items.

ЗМІСТ

ЗМІСТ	3
ВСТУП	4
РОЗДІЛ 1	6
АНАЛІЗ АКТУАЛЬНИХ СИСТЕМ МОНІТОРИНГУ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ НА LINUX СЕРВЕРАХ	6
1.1 Моніторинг апаратного забезпечення	6
1.2 Сучасні платформи моніторингу апаратного забезпечення	7
1.3 Огляд і вибір оптимальної платформи моніторингу	12
1.4 Вибір візуальної оболонки для відображення даних із системи моніторингу	14
1.5 Висновки до розділу 1	15
РОЗДІЛ 2	16
ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ LINUX СЕРВЕРА НА ОСНОВІ ДИСТРИБУТИВУ DEBIAN	16
2.1 Огляд дистрибутиву Debian	16
2.2 Програми для запуску та доступу до Linux сервера	17
2.2.1 Віртуальна машина VirtualBox для запуску операційної системи Linux	18
2.3 Запуск та налаштування Linux	19
2.4 Висновки до розділу 2	21
РОЗДІЛ 3	22
РОЗРОБКА, НАЛАШТУВАННЯ ТА ПРИНЦИП РОБОТИ СИСТЕМИ МОНІТОРИНГУ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ НА LINUX СЕРВЕРІ	22
3.1 Принцип роботи моніторингової системи	22
3.2 Встановлення системи моніторингу на Linux сервер	23
3.3 Доступ до системи моніторингу	27
3.4 Налаштування системи моніторингу	29
3.5 Налаштування візуального відображення у Grafana	31
3.5.1 Огляд метрик, що відображаються	32
3.6 Alert manager. Сповіщення в telegram bot	34
3.6.1 Створення телеграм бота	35
3.6.2 З'єднання менеджера сповіщень з телеграм ботом та створення сповіщень	36
3.6.3 Створення правил-тригерів для сповіщень	37
3.7 Висновки до розділу 3	38
ВИСНОВКИ	40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	41

ВСТУП

Актуальність роботи: Щорічно кількість сайтів у світі збільшується та виростають вимоги як до самих сайтів так і до самих систем моніторингу. Навіть хвилини простою сайтів приносять великих фінансових збитків. Особливо в період карантинів, багато компаній перейшли на онлайн платформи для розміщення замовлень та перегляду інформації. Щоб запобігти можливим проблемам потрібно мати наглядну інформацію про стан апаратного забезпечення серверу перед очима Level 1 Support спеціаліста. Та відповідно якнайшвидше вжити необхідні дії для забезпечення постійної роботи сайтів.

Мета дослідження: Розробити систему зручного та інформативного моніторингу апаратного забезпечення на Linux сервері.

Завдання дослідження:

- Аналіз предметної області;
- Аналіз недоліків та переваг аналогів;
- Обрати та розробити ефективну систему моніторингу апаратного забезпечення на Linux сервері.

Об'єкт дослідження: апаратне забезпечення Linux сервера.

Предмет дослідження: наявні системи для моніторингу Linux сервера.

Практичне значення одержаних результатів: докладна інформація у вигляді метрик та графіків про апаратне забезпечення за допомогою кількох натисків мишки комп'ютера чи телефона.

Зв'язок роботи з науковими програмами, планами, темами. Наведені у роботі дослідження пов'язані з фундаментальною науково-дослідною роботою «Дослідження та комп'ютерно математичне моделювання складних систем та процесів у науці, освіті та інформаційно-комунікаційній діяльності підприємств» (№ держреєстрації 0116U002394, 2018-2022pp.).

Структура кваліфікаційної роботи: Бакалаврська робота складається із вступу, трьох розділів та висновків до них, списку використаних джерел та двох додатків.

У першому розділі бакалаврської роботи розглянуто, що таке моніторингова система, які задачі можуть бути виконані системою. Розглянуті одні з найпопулярніших моніторингових систем та обрана система найбільш доцільна для вирішення поставленої задачі.

У другому розділі бакалаврської пояснений процес встановлення та налаштування Linux сервера.

У третьому розділі бакалаврської роботи наведена інформація що до встановлення, налаштування та доступу до системи моніторингу апаратного забезпечення на Linux сервері.

Бакалаврська робота включає в себе 46 сторінок 31 рисунок і список літератури із 44 джерел.



РОЗДІЛ 1

АНАЛІЗ АКТУАЛЬНИХ СИСТЕМ МОНІТОРИНГУ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ НА LINUX СЕРВЕРАХ

1.1 Моніторинг апаратного забезпечення

Усі проекти з часом раніше або пізніше потрапляють із проблемою до рук фахівця підтримки, системного адміністратора чи розробника. Саме тому кожний сервер має бути забезпечений необхідною підтримкою зі сторони людини та бути постійно під пильним наглядом спеціаліста або автоматизації. Це необхідно для забезпечення стабільної та без перебівної роботи серверу та відповідно наявних на ньому ресурсів [1]. Оскільки стабільність це основа успішного бізнесу для будь-якої сучасної компанії.

Для забезпечення такої постійності розроблені десятки різних систем моніторингу в кожній з яких є свої переваги та недоліки. Сукупність цих факторів і визначає ефективність моніторингу серверу. Цей фактор один із основних чому компанії готові вкладати величезні кошти на розробку та інтегрування таких систем моніторингу для своїх проектів.

Взагалі моніторинг додатків та серверів важлива частина сучасного світу системного адміністратора. Чим більше автоматизована система тим простіше працювати з нею. Звісно хочеться пильнувати одразу все завантаження центрального процесору, оперативну пам'ять, дисковий простір, а в разі проблем отримувати одразу сповіщення про них.

Моніторинг стану апаратної інфраструктури корисний через ряд причин [2], а саме:

- Запобігання проблемам пов'язаним з продуктивністю через можливі проблеми з апаратним забезпеченням.
- Запобігання так званим “downtime” періодам недоступності системи.
- Оптимізація використання наявних апаратних ресурсів системи.
- Зручна система відображення всіх необхідних метрик сервера для стабільної роботи.
- Аналіз періоду не доступності серверу та його активності
- Налаштування сповіщень про критичні показники метрик серверу

Потрібно визначити що ж таке апаратне забезпечення на Linux сервері [3]. Сюди входять різні електронні схеми, що реалізовані у вигляді різних електронних пристроїв та приладів. Простіше кажучи, це те що відноситься до фізичних пристроїв серверу, а саме процесор(CPU), ОЗП(RAM), дисковий

простір, швидкість запису/зчитування даних жорсткого диску (I/O), мережеві порти тощо.

Таким чином система моніторингу має збирати та відображати у вигляді числових показників, графіків, діаграм тощо інформацію про фізичні пристрої сервера.

1.2 Сучасні платформи моніторингу апаратного забезпечення.

Система моніторингу апаратного забезпечення – це програми, платформи, засоби, основною задачею, яких є збір даних від фізичних пристроїв на сервері та відображення цих даних в інтерфейсі, який буде структурований та зрозумілий користувачу. Ця інформація є потрібною для вчасного реагування на проблеми, що виникають на сервері та швидке виправлення цих проблем.

Вимоги до моніторингових систем щороку виростають, оскільки більше даних потрібно обробляти, інтегрувати нові можливості, поліпшувати автоматизації, адже конкуренти не стоять на місці і кожен розробник хоче зайняти свою долю ринку.

Тому якість, швидкість роботи, простота налаштування та інтерфейсу, ефективність – це основні цілі розвитку сучасних моніторингових систем. Те як якісно всі ці дії виконує система і визначає її ефективність.

Існують сотні моніторингових систем із різними можливостями. Далі будуть розглянуті одні з найбільш популярних моніторингових систем: Prometheus, Kibana, Zabbix, Datadog, Splunk, New relic та Nagios [4]. Усі системи є одночасно подібними та кардинально різні. Своя специфіка роботи, орієнтованість на хмарні застосунки або локальні сховища даних, різні методи збереження та передачі даних тощо.

Prometheus – це система з відкритим кодом, яка набула популярності в останні роки. Основною перевагою якої є потужна модель даних і мова запитів. Це вузько направлена система, яка виконує одну задачу і робить це добре.

Швидка система моніторингу на основі метрик, яку використовують десятки тисяч компаній світу. Prometheus включає в себе інформаційну панель та оповіщення. Вільне редагування відкритого коду. Збір показників із сторонніх систем у яких є експортери [5].

Досить часто використовується в поєднанні з веб-додатком Grafana – платформи для аналітики та інтерактивного представлення інформації, яка теж є платформою з відкритим кодом.

Систему можна швидко та легко підняти та підтримувати моніторинг досить часто змінних і важко організованих систем.

Kibana – це інструмент візуалізації та вивчення даних, які використовуються для таких задач, як аналіз журналів логів і часових проміжків, моніторинг додатків і наявних процесів. Система пропонує потужні та прості у використанні можливості: гістограми, лінійні графіки, кругові діаграми (рис. 1.1). Окрім цього, інструмент забезпечує безпосередню взаємодію з Elasticsearch, популярною аналітичною та пошуковою системою, за рахунок чого Kibana постає в ролі візуалізації даних, які зберігаються в Elasticsearch [6].

З не давніх пір інструмент постачається як ліцензійний, тому відкрито використовувати його є досить проблематично. Хоча є певні ліцензії Elastic або SPL, які дозволяють розробникам вносити зміни до коду програми.

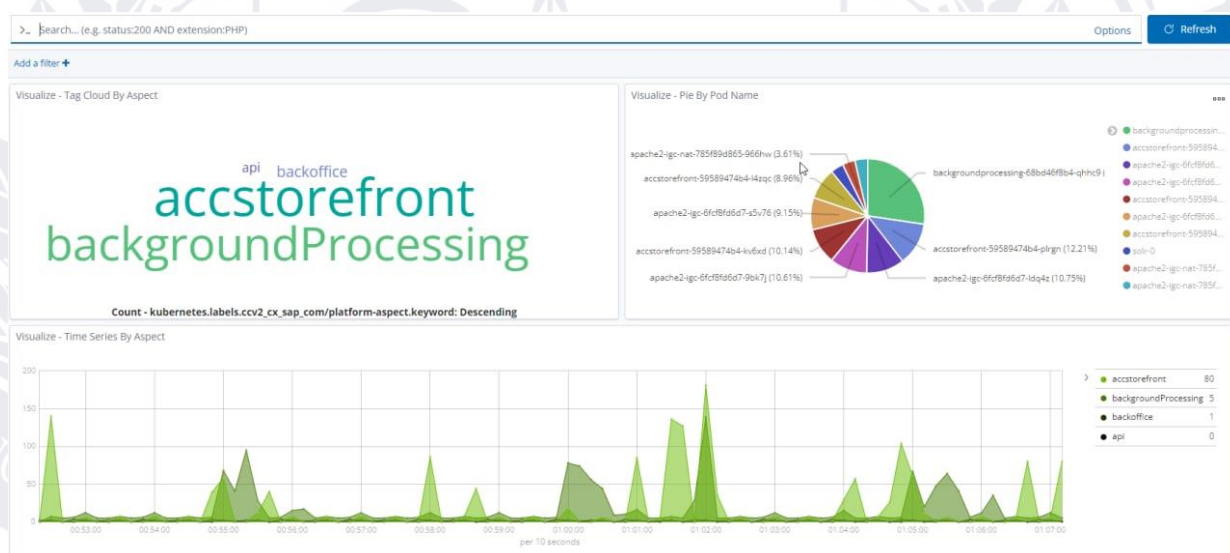


Рисунок 1.1 – Kibana інтерфейс моніторингової системи

Zabbix – пропонує корисну інформацію про продуктивність і проблеми інфраструктури. Також надає можливість покращення системи моніторингу за допомогою різних потужних влаштованих інструментів.

Zabbix містить інструменти створення елементів та тригерів для різних типів моніторингу, створення шаблонів і використання власної проксі Zabbix. Можливе використання Zabbix API для налаштування та ефективного керування серверами та самим сервером Zabbix. Наявна відкрита база для пошуку готових рішень поширених та не дуже проблем із якими можна зіткнутись під час повсякденного використання Zabbix [7].

Функціонал включає в себе загальні перевірки поширених сервісів, враховуючи СУБД, SSH, telnet, NTP, POP, SMTP, FTP тощо. Якщо стандартних

інструментів не достатньо для поставлених задач, то їх можна самостійно змінити або використовувати додатки через API.

Загалом, Zabbix містить всі необхідні та ефективні інструменти для потужної системи моніторингу, яка може бути заточена під індивідуальні потреби кожного проекту за допомогою використання ключових можливостей Zabbix, наприклад інформація часу відгуку сайту(рис. 1.2). Хоча це універсальна система моніторингу ресурсів, апаратної та програмної частини серверу, яка є безкоштовною.

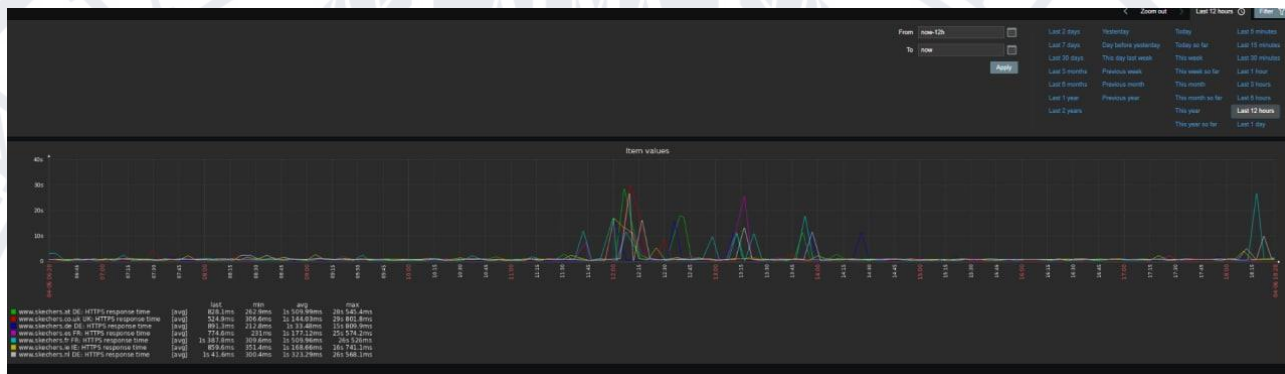


Рисунок 1.2 – Час відгуку сайтів у Zabbix

Datadog – хмарна моніторингова система в онлайн режимі та оцінки продуктивності мережі, дій користувачі, процесів та додатків. Система є важливою платформою моніторингу та безпеки для хмарних додатків. Об'єднує наскрізні трасування, метрики та журнали логів, щоб зробити програми, інфраструктуру та інші різні служби повністю видимими(рис 1.3). Ці можливості системи допомагають компаніям захистити свої проекти, уникнути простоїв та можуть гарантувати, що клієнти отримають відповідний рівень обслуговування [8].

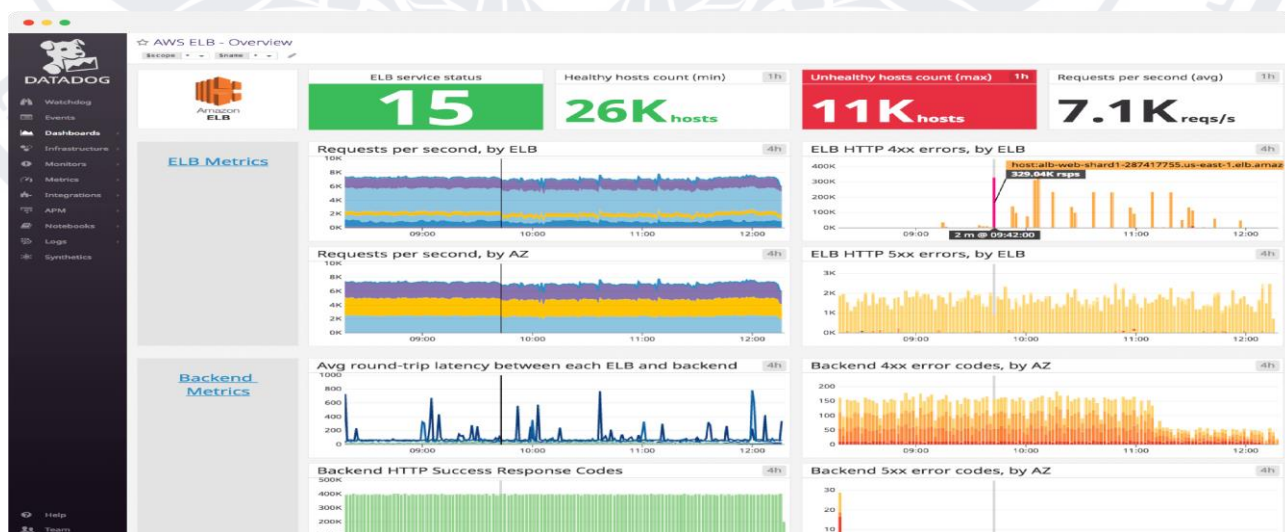


Рисунок 1.3 – Datadog інтерфейс

Datadog поєднує дані по всьому стеку за допомогою більш як 400 інтеграцій для усунення проблем, відправки сповіщень та побудування зв'язних графів. Систему можна використовувати у якості єдиного інструменту для усунення недоліків, оптимізації продуктивності і кооперативної роботи між різними компаніями.

Насамперед це платформа моніторингу для великомасштабних додатків. Оскільки система поєднує в собі моніторинг інфраструктури, моніторинг продуктивності додатків.

Splunk – це моніторингова платформа з широким асортиментом у використанні. Починаючи, від моніторингу інфраструктури, відправлення покращених сповіщень та покращені засоби безпеки [9].

На сьогоднішній момент система дуже поширена в США та Європі. Поступово виходить на нові ринки збільшуючи свою популярність. Головна особливість платформи, що вона може працювати з даними практично з будь-яких джерел, а тому способи використання є досить широкі.

Splunk зазвичай розбирає дані на “поля” та значення після чого обробляє їх. Обробка даних здійснюється за допомогою SPL запитів(спеціальна мова розроблена Splunk) за допомогою, яких надається можливість побудування різних вибірок даних, таблиць, сортувати дані за вказаним параметром, будувати звіти та головне створювати дашборди з необхідною інформацією як на рисунку(рис. 1.4).

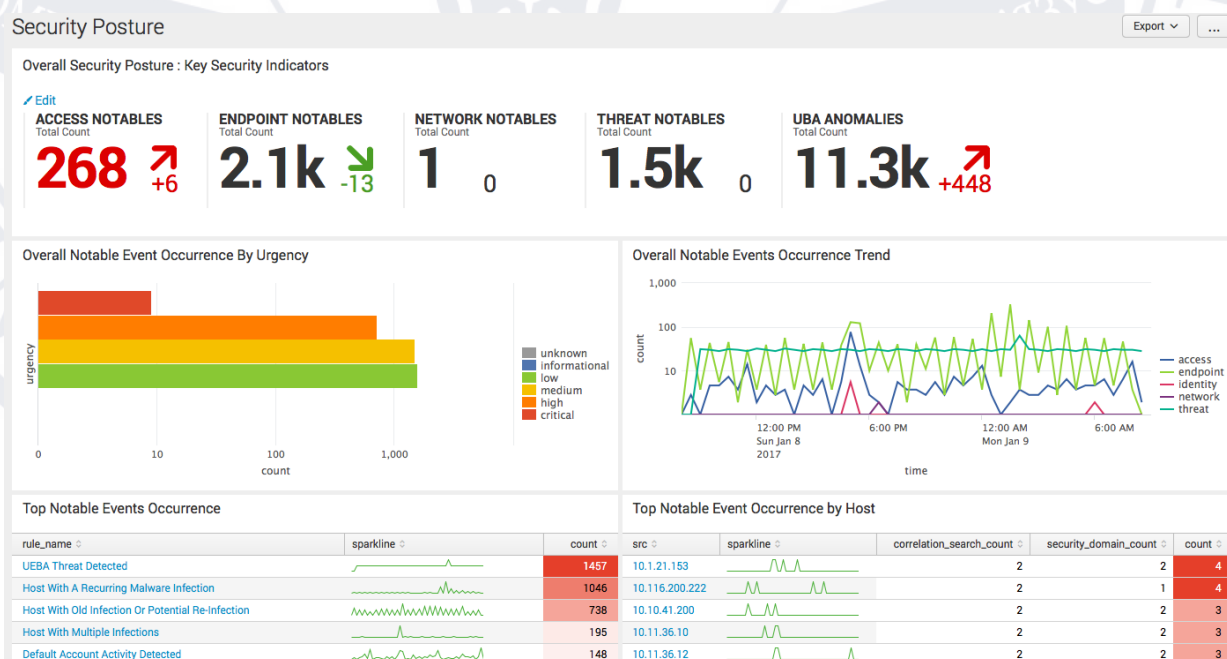


Рисунок 1.4 – Splunk дашборд

New relic – це хмарна система для моніторингу CPU, Memory, дискового простору та інших апаратних компонентів. Система збирає всю інформацію у хмарину там і зберігає, що дає змогу не використовувати власну фізичну пам'ять для даних. Зберігаються метрики, трасіровки, журнали логів і все в одному єдиному місці(рис. 1.5).

Надає можливість побудування запитів для пошуку потрібних даних із зібраної інформації, відправлення сповіщень тривоги та інформаційну панель – дашборд для відображення інформації у візуальній оболонці. Можливість кореляції даних у всьому стеці. До того ж серед особливостей та налаштована система допомоги штучного інтелекту у всьому процесі роботи з New relic [10].

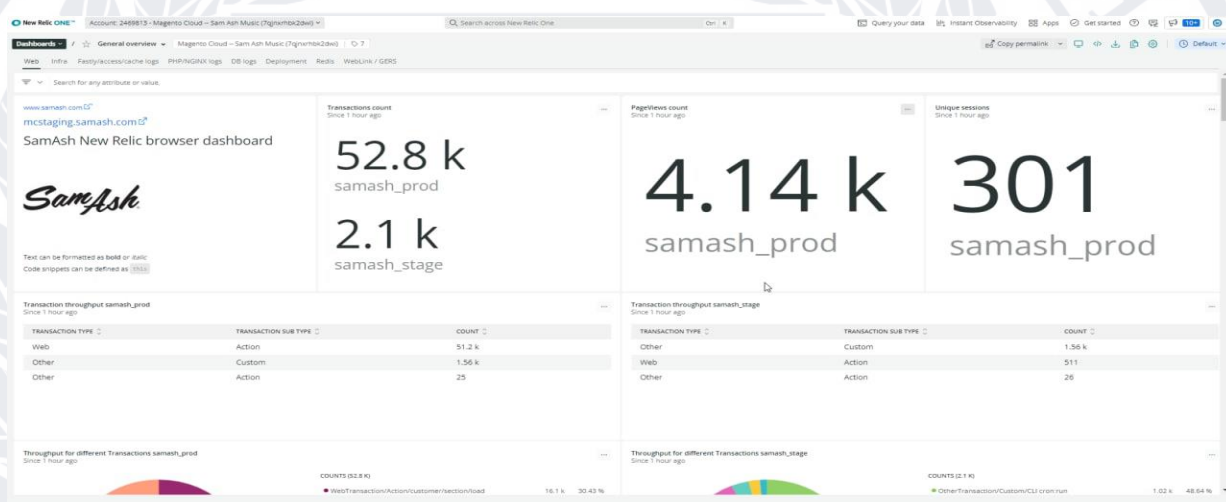
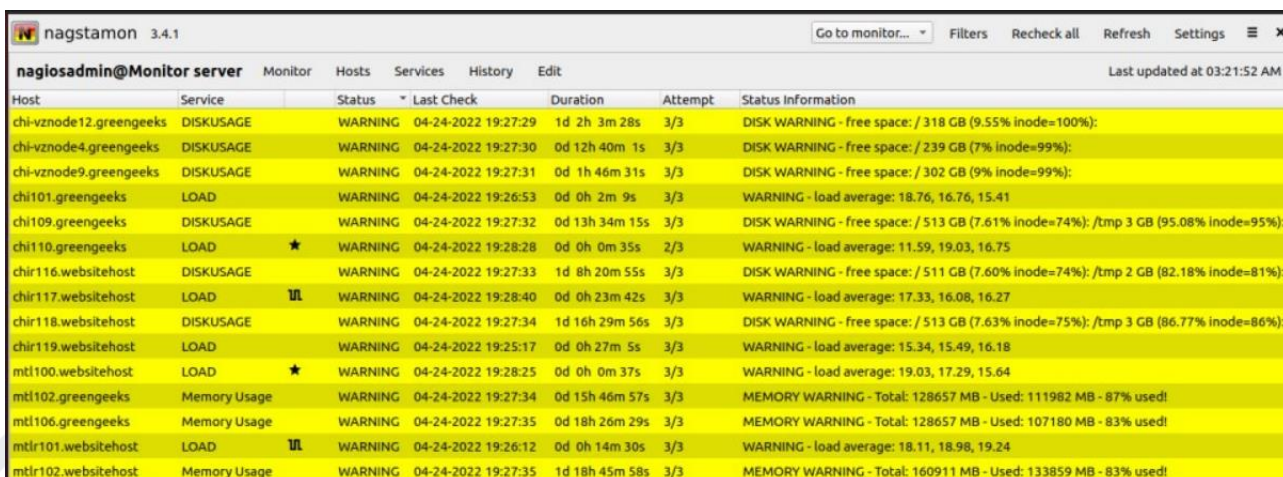


Рисунок 1.5 – Загальна інформація New Relic моніторингу

Nagios - ядро яке використовують найчастіше з утилітою Nagstamon. Це система з відкритим кодом, яка є дуже гнучкою в налаштуваннях та конфігурації, але дещо застаріла в порівнянні з більш новими системами моніторингу.

Система має можливість моніторингу мережевих послуг(SMTP,HTTP,PING,POP3 тощо). Моніторинг метрик таких як CPU, Memory usage, I/O usage тощо(рис. 1.6). Легкий дизайн та інтерфейс системи, що знижують необхідний рівень знань для конфігурації системи. Одночасні паралельні перевірки серверів [11]. Система сповіщень для різних типів помилок має вигляд як на рисунку нижче

Підключення віддаленого моніторингу до серверів виконується за допомогою шифрованих тунелів SSH або SSL.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
chi-vzn02.greengeeks	DISKUSAGE	WARNING	04-24-2022 19:27:29	1d 2h 3m 28s	3/3	DISK WARNING - free space: / 318 GB (9.55% inode=100%):
chi-vzn04.greengeeks	DISKUSAGE	WARNING	04-24-2022 19:27:30	0d 12h 40m 1s	3/3	DISK WARNING - free space: / 239 GB (7% inode=99%):
chi-vzn09.greengeeks	DISKUSAGE	WARNING	04-24-2022 19:27:31	0d 1h 46m 31s	3/3	DISK WARNING - free space: / 302 GB (9% inode=99%):
chi101.greengeeks	LOAD	WARNING	04-24-2022 19:26:53	0d 0h 2m 9s	3/3	WARNING - load average: 18.76, 16.76, 15.41
chi109.greengeeks	DISKUSAGE	WARNING	04-24-2022 19:27:32	0d 13h 34m 15s	3/3	DISK WARNING - free space: / 513 GB (7.61% inode=74%): /tmp 3 GB (95.08% inode=95%):
chi110.greengeeks	LOAD	WARNING	04-24-2022 19:28:28	0d 0h 0m 35s	2/3	WARNING - load average: 11.59, 19.03, 16.75
chir116.websitehost	DISKUSAGE	WARNING	04-24-2022 19:27:33	1d 8h 20m 55s	3/3	DISK WARNING - free space: / 511 GB (7.60% inode=74%): /tmp 2 GB (82.18% inode=81%):
chir117.websitehost	LOAD	WARNING	04-24-2022 19:28:40	0d 0h 23m 42s	3/3	WARNING - load average: 17.33, 16.08, 16.27
chir118.websitehost	DISKUSAGE	WARNING	04-24-2022 19:27:34	1d 16h 29m 56s	3/3	DISK WARNING - free space: / 513 GB (7.63% inode=75%): /tmp 3 GB (86.77% inode=86%):
chir119.websitehost	LOAD	WARNING	04-24-2022 19:25:17	0d 0h 27m 5s	3/3	WARNING - load average: 15.34, 15.49, 16.18
mtl100.websitehost	LOAD	WARNING	04-24-2022 19:28:25	0d 0h 0m 37s	3/3	WARNING - load average: 19.03, 17.29, 15.64
mtl102.greengeeks	Memory Usage	WARNING	04-24-2022 19:27:34	0d 15h 46m 57s	3/3	MEMORY WARNING - Total: 128657 MB - Used: 111982 MB - 87% used!
mtl106.greengeeks	Memory Usage	WARNING	04-24-2022 19:27:35	0d 18h 26m 29s	3/3	MEMORY WARNING - Total: 128657 MB - Used: 107180 MB - 83% used!
mtlr101.websitehost	LOAD	WARNING	04-24-2022 19:26:12	0d 0h 14m 30s	3/3	WARNING - load average: 18.11, 18.98, 19.24
mtlr102.websitehost	Memory Usage	WARNING	04-24-2022 19:27:35	1d 18h 45m 58s	3/3	MEMORY WARNING - Total: 160911 MB - Used: 133859 MB - 83% used!

Рисунок 1.6 – Моніторинг системою Nagstamon

Оскільки ядро з відкритим кодом і написане досить давно існує багато доповнень до основного функціоналу. Одні з найпопулярніших:

- NRPE - дозволяє запускати плагіни для перевірки навантаження процесору(ядер), оперативної пам'яті, дискового простору на віддаленому сервері Linux.
- NSCA - дозволяє надсилати результати пасивних перевірок простору з віддалених Linux хостів демону Nagios на сервері моніторингу.
- NDOUtils – дозволяє зберігати всю інформацію програми Nagios у єдиній базі MySQL.

1.3 Огляд і вибір оптимальної платформи моніторингу

Оскільки необхідно створити систему моніторингу апаратного забезпечення на власному Linux сервері – буде обрано оптимальний варіант, який дозволить вирішити поставлену задачу.

Головними критеріями при виборі буде відносна простота налаштування та водночас потужна платформа з широким спектром інструментів. Серед важливих параметрів також стабільність та надійність платформи, яка може чітко та надійно забезпечити безперервність роботи Linux сервера. Не мало важливий фактор це доступність платформи. Також платформа має бути націлена на збір даних саме апаратного забезпечення з можливістю збору основних показників (CPU, RAM, I/O...) серверу [12].

На основі аналізу у підрозділі 1.2 та зазначених вище критеріїв можна відокремити дві основні платформи, а саме Prometheus та New relic.

На основі порівняння даних платформ New relic та Prometheus для виконання поставленої задачі більше підходить платформа Prometheus. Згідно таблиці 1.1.

Таблиця 1.1 – порівняння моніторингових систем Prometheus та New relic.

Параметр порівняння	Prometheus	New relic
Простота налаштування	Швидко та легко розгорнути	Досить складна через великий набір інструментів (наявний штучний інтелект- помічник)
Спектр додатків та потужність інструментів	Широкий спектр можливостей. Достатня кількість для збору основних даних апаратного забезпечення, можливість заточення системи під власні конкретні задачі	Широкий вибір інструментів, побудування запитів та аналізу даних, відображення даних у дашборді та зберігання даних в хмарині.
Збір даних апаратного забезпечення	+	+
Стабільність та надійність системи	Власний сервер, де контролюється та налаштовується все самостійно	Хмарина. В основному залежить від команди New Relic.
Доступність	Відкритий код у вільному доступі	Ліцензійна щомісячна передплата

Серед основних переваг це відкритий код, який можна вільно редагувати та налаштовувати систему моніторингу для власних цілей, розгортання системи на власному сервері, що в сукупності дає можливість націлити систему саме на моніторинг даних апаратного забезпечення на Linux сервері. Загалом Prometheus забезпечений усім необхідним асортиментом інструментів для побудування ефективної системи моніторингу.

До недоліків New Relic у випадку поставленої задачі можна віднести орієнтованість на роботу з логами та відображенням інформації орієнтованої на

цих логах. Інформація різноманітна, а в поставленій задачі орієнтованість на моніторинг апаратного забезпечення на Linux сервері. Тому використання NewRelic буде не доцільним, оскільки більшість інструментів платформи не будуть використані та можуть викликати проблеми при налаштуванні основного моніторингу апаратного забезпечення.

1.4 Вибір візуальної оболонки для відображення даних із системи моніторингу

Зібрати дані про апаратне забезпечення серверу не так важко, але ще потрібно представити ці дані в доступному та легкому вигляді для звичайного користувача. Саме тому використовують різні графічні оболонки для цього з влаштованими інструментами для налаштування відображення. Зазвичай відображають дані у вигляді зручних та інформативних таблиць “dashboard” на яких виведена уся необхідна інформація [13].

Для роботи з Prometheus найчастіше використовують візуальну оболонку під назвою Grafana. Це також платформа з відкритим кодом для візуалізації, моніторингу, аналізу даних, Цей інструмент у поєднанні з Prometheus і являє собою так звану систему моніторингу [14].

Серед можливостей Grafana слід виділити створення дашбордів із різними панелями, кожна панель відображає якусь певну характеристику/показники за деякий час періоду, який налаштовується самостійно. Кожний дашборд універсальний та індивідуальний тому його можна налаштувати під конкретні задачі або проект самостійно, враховуючи будь-які потреби поставленої задачі.

Для кожного джерела даних у Grafana передбачено редактор запитів та спеціальний синтаксис, який можливо налаштовувати. Завдяки цьому і можливе налаштування дашбордів під конкретні індивідуальні запити та задачі.

Платформа написана на мові програмування Go, так само як Prometheus, разом із потужною системою прикладного інтерфейсу (API).

До переваг можна віднести конкретні рішення задач. Тобто якщо вся необхідна інформація не може бути виведена на один екран – вона розбивається на декілька розділів, які є зручними для перегляду. Підрозділи можуть відображатись автоматично та періодично. Період зміни розділів теж може бути налаштованим, тому ніяка важлива інформація не буде пропущена.

Доволі зрозуміле та інформативне відображення даних із графіками, відсотками, значеннями. Також наявне кольорове відображення даних, що дає

можливість зрозуміти у якому стані перебуває той чи інший показник, де зелений/синій колір є нормою, а червоний колір – критичним показником (рис. 1.7).



Рисунок 1.7 – Відображення даних зібраних Prometheus

1.5 Висновки до розділу 1

У даному розділі було детально розглянуто та обґрунтовано, що таке системи моніторингу апаратного забезпечення на Linux сервері. Після проведеного аналізу існуючих платформ - обрано платформу та графічну оболонку. На основі яких буде розроблена система моніторингу апаратного забезпечення на Linux сервері відповідно до поставленої задачі. Наступний розділ буде присвячений розгортанню та налаштуванню Linux сервера на основі дистрибутива Debian.

РОЗДІЛ 2

ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ LINUX СЕРВЕРА НА ОСНОВІ ДИСТРИБУТИВУ DEBIAN

2.1 Огляд дистрибутиву Debian

GNU/Linux – це сімейство Unix подібних операційних систем на базі ядра Linux та програм GNU [15].

Операційна система – це основний комплекс програмного забезпечення, що виконує управління апаратного забезпечення на комп'ютері, забезпечує логістику роботи всіх процесів і допомагає звичайному користувача взаємодіяти з комп'ютером за допомогою інтерфейсу [16].

Для розробки системи моніторингу Linux сервера насамперед треба, щоб був працюючий сервер із якого ці дані будуть збиратись. На даний момент часу існують десятки різних дистрибутивів Linux із своїми нюансами та можливостями. У випадку поставленої задачі вірним варіант буде використати Linux сервер на основі дистрибутиву Debian.

Сервер – це комп'ютер призначений для обробки запитів та передачі даних до інших комп'ютерів за допомогою глобальної мережі інтернет або локальної мережі. Існує чимало різних типів серверів, кожен із яких добре виконує свої поставлені задачі [17].

У рамках поставленої задачі доцільно використати веб-сервер. Веб-сервер показує сторінки та запускає програми за допомогою браузера. Тобто сервер до якого підключається браузер – це веб-сервер, який і обробляє запити від браузера та надає сторінки чи запускає програми.

Debian – це система яка поєднує в собі найновіші розробки з максимальною ефективністю. На даний момент – це один із найбільш надійних дистрибутивів Linux. У його арсеналі одні з найкращих та найновіших доступних пакетів, велика кількість користувачів та розробників, що працює на новими версіями та високий стандарт якості й безпеки [18].

Чому саме Debian:

Логічна та проста файлова структура, де файли користувача розміщені в /usr/bin , файли конфігурації в /etc/ , а логи там де мають бути в /var/log.

- Надійність. Про систему насправді не так багато інформації в мережі, скоріш за все через те, що вона не часто видає помилки і користувачам не треба проводити десятки годин у пошуках рішення.
- Підтримка різних пристроїв, можливістю автоматичного налаштування та легкого налаштування.
- Вільна для поширення система. Всі пакети можна вільно використовувати без будь-яких ліцензій
- Систему потрібно раз налаштувати і вона буде виконувати свою роботу добре та стабільно

2.2 Програми для запуску та доступу до Linux сервера

Поширена проблема серед користувачів платформи Windows, оскільки постає питання, а як запустити Linux? Серед очевидних використання окремої робочої станції (ПК) із системою OS Linux, але якщо немає такої можливості будуть розглянуті альтернативи.

У сучасному світі є безліч можливостей для запуску Linux. Зокрема встановлення Linux на комп'ютері паралельно операційній системі Windows та загрузка систем за допомогою вибору в завантажувачі Grub, проте цей варіант не доцільний у поставленій задачі роботи, оскільки потрібна постійна доступність Windows системи. Можна виділити два основні варіанти це підсистема WSL або використання віртуальних машин (VirtualBox, VMWare...). Набуває популярності використання хмарних застосунків таких як AWS(Amazon web services) або Salesforces тощо [19].

Насправді не так важливо який саме спосіб обрати, оскільки у всіх варіантах кінцевий результат буде один – Linux сервер.

Підсистема Windows для Linux (WSL) дозволяє запускати середовище GNU/Linux із можливістю використання більшості команд через більшість програм командного рядка, службових програм або безпосередньо з системи Windows без необхідності зміни операційної системи. У Microsoft store доступні майже всі дистрибутиви для завантаження та вільного доступу до них через стандартний командний рядок Windows, який можна відкрити за допомогою команди cmd у пошуку по системі.

Основною, а точніше головною проблемою є відсутність можливості встановлення серверних дистрибутивів по типу Debian server. Тому доцільно буде використати віртуальну машину, щоб мати можливість користуватись усім асортиментом функцій та можливостей системи Linux.

Хоча звісно можна орендувати або купити місце на готовому сервері де система буде одразу налаштована для роботи із нею, проте навіть у такому випадку скоріше за все знадобиться термінал із SSH доступом до серверу.

SSH – це протокол віддаленого доступу або адміністрування розроблений для управління операційними системами. Застосування цього протоколу гарантує використання різних алгоритмів для шифрування даних та передачі даних по захищеному каналі. По суті це дає змогу безпечно працювати в будь-якому середовищі [20].

Для підключення до сервера буде використана проста та вільна в доступі утиліта Putty. Це програма підключення до серверу через безпечний зв'язок SSH [21]. Програма необхідна для виконання команд у системі Linux та зручного перегляду командного рядка системи. Все що потрібно для доступу це піднятий SSH сервіс на сервері, виділений порт для доступу (стандартний 22) та IPv4 адреса сервера.

2.2.1 Віртуальна машина VirtualBox для запуску операційної системи Linux

VirtualBox – це програмний засіб для віртуалізації різних операційних систем точніше він симулює справжній комп'ютер, що дає змогу користувачам встановлювати, запускати та використовувати різні операційні системи. Простіше кажучи – це комп'ютер в комп'ютері [22].

VirtualBox розповсюджується вільно, та може бути завантажена з офіційного сайту. Також потрібен сам дистрибутив Linux, який буде запускатися з віртуальної машини. Таким способом із комп'ютера з операційною системою Windows можливо запустити віртуальний комп'ютер із операційною системою на основі Linux дистрибутиву.

Щоб запустити ОС Linux спершу необхідно створити нову віртуальну машину за допомогою кнопки інтерфейсу програми “Створити”. Далі треба обрати тип системи: Linux, версію: Debian 64 bit [23](відповідно до необхідної розрядності), вибрати на локальному диску папку виділену під віртуальну машину та назвати її.

Звичайно існують певні недоліки даного методу обмеженість в ресурсах, необхідність постійно зберігати дані, а краще робити резервні копіювання, щоб не втратити дані про налаштовану машину.

Віртуальна машина появиться у головному вікні програми(рис. 2.1), але цього не достатньо для запуску Linux системи.

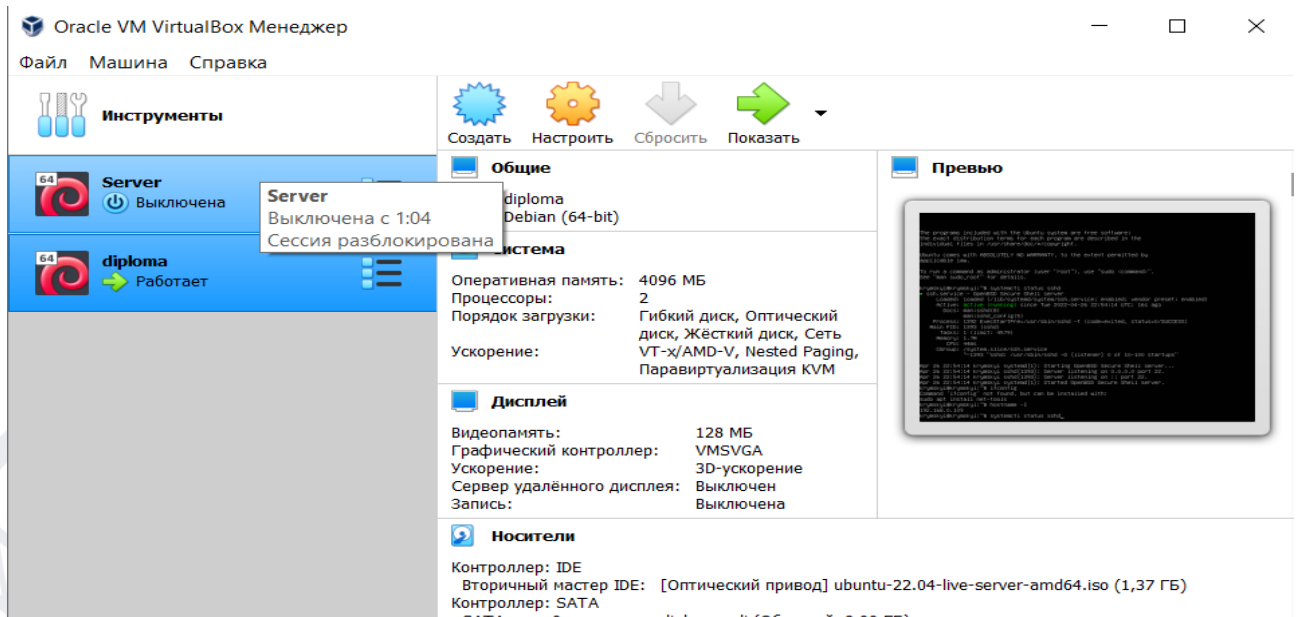


Рисунок 2.1 – інтерфейс VirtualBox

Потрібно ще вибрати образ диска з якого буде зчитуватись Linux та прокинути порти для подальшого доступу до сервера через SSH протокол тощо... Натиснувши кнопку налаштування треба перейти у розділ “Накопичувачі” та знайти меню Контролер: IDE під яким буде іконка диска, натиснувши на нього появиться кнопка + , що дає змогу вибрати образ диска Debian-live-server-amd64.iso. Тепер віртуальна машина при запуску буде використовувати дані з цього диска та запускати систему Linux. Щоб прокинути порти між системою Windows та віртуальною машиною треба перейти в розділ “Мережа” після чого достатньо змінити тип підключення з “NAT” на “Мережевий міст”. Таким чином віртуальна система буде використовувати ті самі порти та IP-адреси для доступів, як і система хазяїн Windows.

Тепер віртуальна машина готова для запуску Linux системи.

2.3 Запуск та налаштування Linux

Оскільки все готово для запуску серверу у Virtual Box треба запустити створену машину та провести налаштування. Оскільки використовується дистрибутив для серверу – візуальний інтерфейс не буде доступний, а все буде виконуватись із командного рядка.

Після запуску появиться меню Install Debian server, яке потрібно натиснути - після чого почнеться автоматичне встановлення операційної системи. Після встановлення потрібно вибрати мову системи - краще використовувати англійську. Управління здійснюється за допомогою стрілок клавіатури, клавіш “Enter” та “Space”. Далі вибрати розкладку клавіатури та

вибрати повну інсталяцію Debian Server системи. Наступним пунктом буде налаштування мережі, але оскільки у розділі 2.2.1 вже був налаштований мережевий міст, точ майбутній сервер буде брати всі необхідні дані від батьківської системи Windows і додаткові налаштування в цьому етапі не потрібні. Якщо є наявна проксі адреса можна ввести її в наступному етапі або пропустити етап якщо доступ по IP-адресі є доцільним рішенням. Наступними етапами буде розмітка диску, можна залишити за замовчуванням та створення аккаунта з правами root доступу, тому важливо не забути пароль для подальшої роботи з сервером(рис. 2.2).

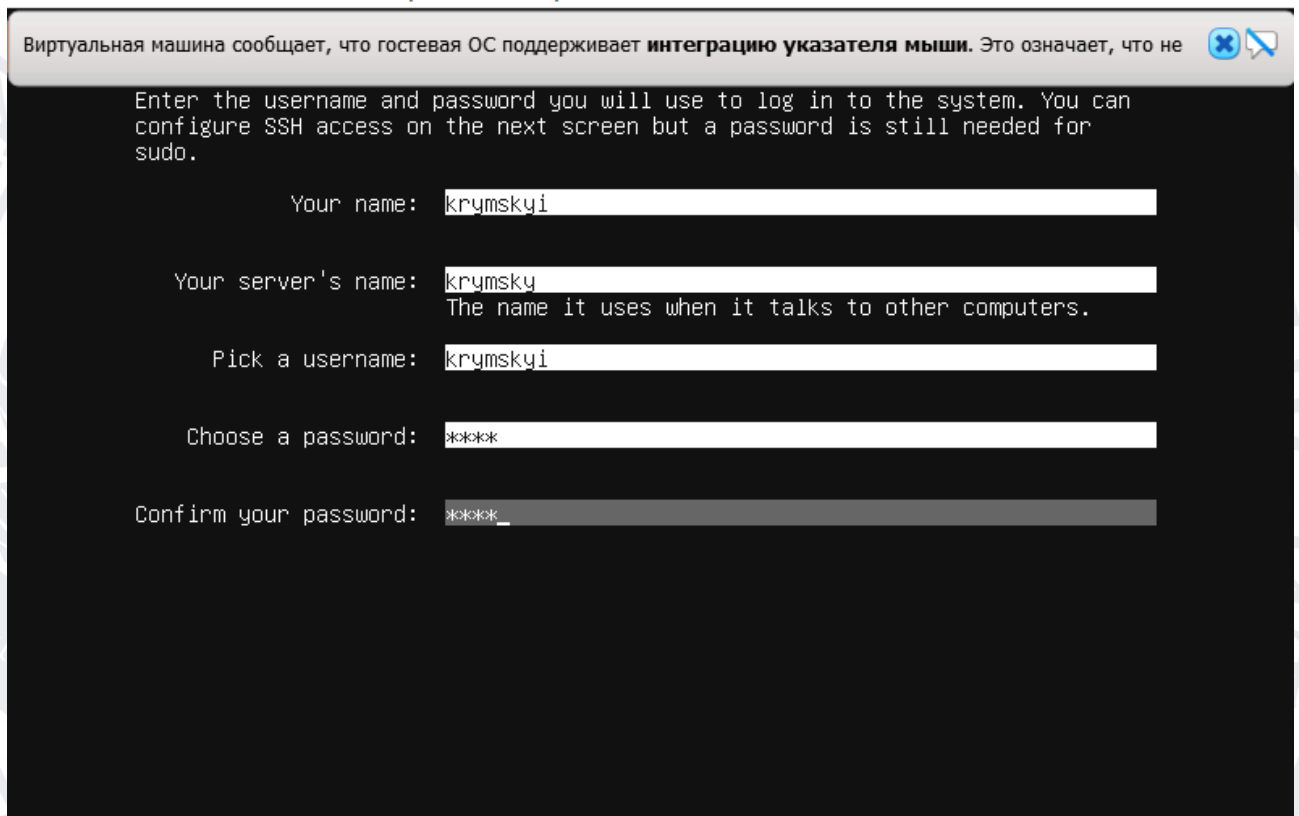


Рисунок 2.2 – створення аккаунту з root доступом

Останнім пунктом обов'язково вибрати встановлення OpenSSH сервера та запустити процес встановлення операційної системи. Після встановлення натиснути кнопку перезапуску системи після чого сервер на Linux буде працювати.

За допомогою команд `apt install ssh` та `systemctl status ssh` встановлюється SSH сервер та перевіряється чи активний сервіс SSH [24].

За допомогою команди `sudo apt install net-tools` встановлюються пакети для перегляду мережевих налаштувань. Команда `ifconfig`(рис. 2.3) виводить на екран інформацію про наявну мережу. Серед деталей необхідно знайти основний інтерфейс та дізнатись inet IPv4 адресу сервера на рисунку показано,

що 192.168.0.109 IP адреса серверу.

```
krymskyi@krymskyi:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.109 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:feb8:60e8 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:b8:60:e8 txqueuelen 1000 (Ethernet)
    RX packets 856 bytes 319197 (319.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 699 bytes 127390 (127.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 120 bytes 10406 (10.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 120 bytes 10406 (10.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Рисунок 2.3 – вивід команди ifconfig

Відтепер до серверу може бути здійснений віддалений доступ по SSH через утиліту Putty (рис. 2.4)

Рисунок 2.4 – доступ по SSH через утиліту Putty

2.4 Висновки до розділу 2

У даному розділі були розглянуті процеси встановлення та налаштування Linux сервера на основі дистрибутиву Debian. Обґрунтовані вибір дистрибутиву Debian та його основні переваги. Детально розглянуті технології розгортання власного сервера на основі операційної системи Linux. Та самі процеси створення серверу в додатку віртуалізації Virtual Box і встановлення операційної системи Linux. У наступному розділі буде розглянуто більш детально принцип роботи моніторингової системи на платформах Prometheus і Grafana, встановлення платформ на сервер та їх налаштування.

РОЗДІЛ 3

РОЗРОБКА, НАЛАШТУВАННЯ ТА ПРИНЦИП РОБОТИ СИСТЕМИ МОНІТОРИНГУ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ НА LINUX СЕРВЕРІ

3.1 Принцип роботи моніторингової системи

Перш за все варто відзначити, що Prometheus – це не готове рішення вигляду встановив і працює, а складна платформа, яка може працювати з різними механізмами та програмами. Простіше кажучи існує безліч варіантів конфігурацій моніторингових систем, кожна з яких по суті є унікальною.

Щоб правильно налаштувати моніторингову систему потрібно розуміти як вона фундаментально працює для того щоб знати, а що саме налаштовувати. Тому у цьому розділі буде більш детально розглянуто принцип роботи як Grafana взагалі поєднується і працює з Prometheus і що потрібно для успішної роботи системи.

Оскільки платформи вільно розповсюджуються на офіційних джерелах можна знайти посилання на Docker контейнери з фундаментальним кодом платформ [25]. Оскільки це контейнери для роботи з ними зазвичай використовують таку платформу як Docker. Docker – це просто кажучи інструкція для розгортання чогось. Він потрібен для більш ефективного використання системи та наявних ресурсів у ній, швидкого та легкого розгортання готових програмних продуктів, з якими в подальшому досить легко працювати та модернізувати [26].

З рисунку 3.1 видно, що перший етап – це збір метрик із різних процесів системи (робіт) як довготривалих так і короткочасних. За цей етап якраз і відповідає node exporter. Тому у node exporter прописані всі метрики, які збираються на сервері.

Pushgateway - це сервіс куди можна відправляти метрики, коли стандартна pull-модель Prometheus, ще не може бути виконана [28].

Далі вступає в роботу сам Prometheus, який збирає всі дані з Pushgateway в одному місці у так званій TSDB – базі даних часових рядів і зберігає їх на фізичному накопичувачі HDD або SSD. Для координації даних у цій базі використовується технологія Kubernetes. Для того, щоб витягнути дані з Prometheus сервера використовується технологія HTTP – протокол прикладного рівня передачі даних. За допомогою HTTP є можливість передачі даних для візуалізації в Grafana та доступ до цих даних за допомогою API клієнтів. Також у Grafana реалізований інструмент сповіщень, який витягує дані по протоколу

HTTP із Prometheus сервера та надсилає сповіщення за допомогою різних способів(Slack, Email,Telegram тощо).

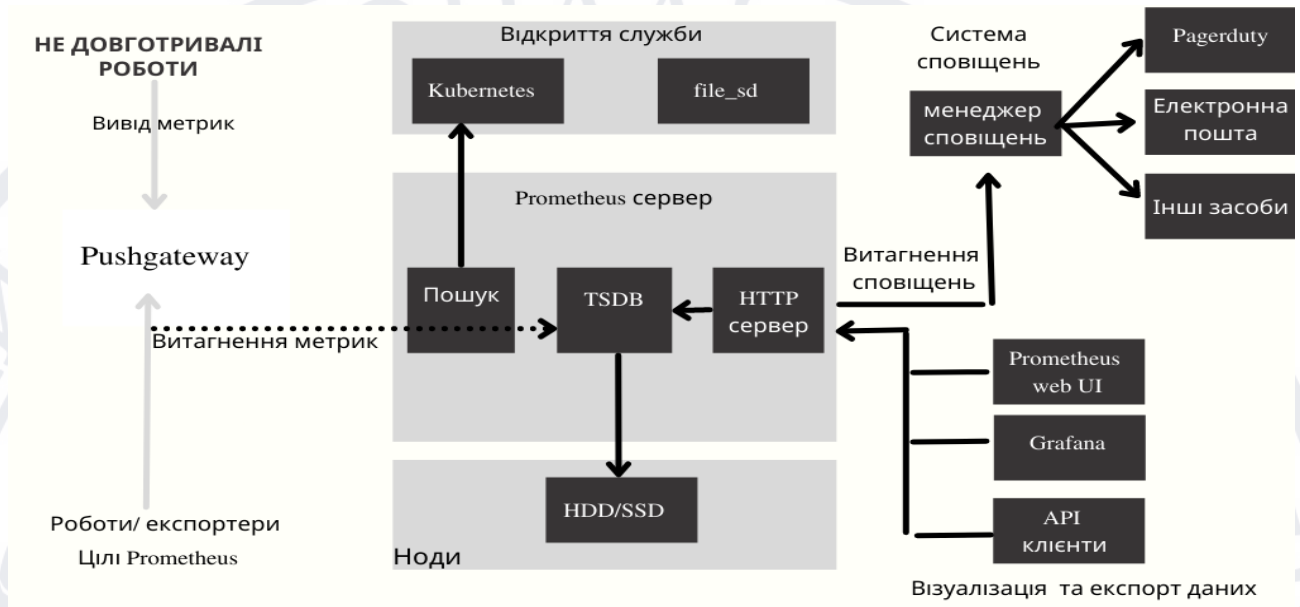


Рисунок 3.1 – схема роботи Prometheus

3.2 Встановлення системи моніторингу на Linux сервер

Перш за все треба підключись до сервера на якому буде встановлена система моніторингу, підключення буде здійснене за допомогою протоколу SSH через утиліту Putty. Для цього потрібен лише IP-адреса хоста та відкритий порт для підключення, а також звісно існуючий користувач із логіном та паролем через, які буде здійснено вхід на сервер.

Першим кроком перед будь-яким виконанням робіт на сервері має бути оновлення джерел пакетів, а тому необхідно виконати команди `apt update` та `apt-get update` (рис. 3.2). Команда синхронізує списки пакетів у системі. Це дозволить використовувати найбільш актуальні дані для встановлення.

```
[sudo] password for krymskiy:
root@krymskiy:/home/krymskiy# apt update
Hit:1 http://ua.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ua.archive.ubuntu.com/ubuntu jammy-updates InRelease [109 kB]
Hit:3 http://ua.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://ua.archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://ua.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [50.7 kB]
Get:6 http://ua.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [63.5 kB]
Fetched 334 kB in 1s (341 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
12 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@krymskiy:/home/krymskiy#
```

Рисунок 3.2 – використання команди apt update

не змогли отримати доступ до контейнерів та все зіпсувати чи використати у власних цілях [30].

Не обов'язково використовувати саме `admin` значення, вони можуть бути довільними. У рамках поставленої задачі використанні значення `admin` для зручності. Оскільки все встановлюється на локальному сервері до якого не буде доступу з глобальної мережі.

Наступним етапом необхідно скопіювати репозиторію `Docker-compose-Prometheus-and-Grafana`. Репозиторій можна скласти як самому оскільки дані технології із відкритим кодом і можна завантажити з офіційних джерел всі файли, а далі загрузити на сервер за допомогою FTP або в GitHub після чого клонувати з GitHub на сервер. Проте цей варіант буде доцільним при бажанні чи потреби специфічного налаштування репозиторію.

Кращим рішенням буде знайти вже готову репозиторію на GitHub, щоб не витрачати час і клонувати репозиторію на сервер. Даний варіант дає змогу значно спростити розгортання системи моніторингу та заощадить час на розгортанні.

`Clone` – це клонування даних, зазвичай репозиторій, які знаходять у відкритому доступі на сервер, локальний комп'ютер тощо [31].

GitHub – одне з найбільш популярних місць збереження репозиторіїв, різних файлів із кодом, програм тощо [32]. Серед ІТ спеціалістів це свого роду Вікіпедія, де завжди можна знайти потрібну інформацію, поділитися даними з знайомими. Оскільки є можливість налаштовувати права доступу до репозиторіїв – це зручний інструмент для розповсюдження або збереження даних.

Щоб зробити клон репозиторію на сервер треба виконати команду `git clone` та посилання звідки клонувати дані(рис. 3.5).

Після чого будуть завантажені на сервер файли розміщені по адресі, яку задали для клонування.

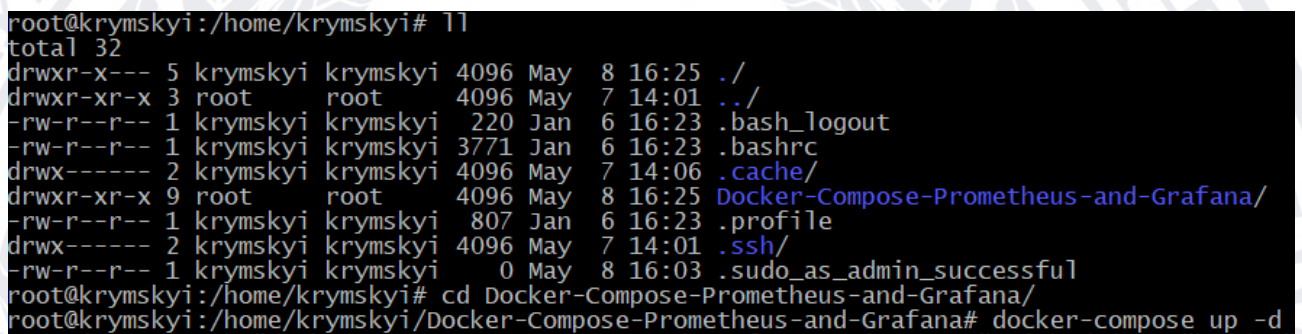
```
root@krymskyi:/home/krymskyi# git clone https://github.com/Einsteinish/Docker-Compose-Prometheus-and-Grafana
Cloning into 'Docker-Compose-Prometheus-and-Grafana'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 40 (delta 1), reused 0 (delta 0), pack-reused 32
Receiving objects: 100% (40/40), 2.90 MiB | 6.46 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

Рисунок 3.5 – клонування репозиторію

Далі треба перейти в папку з встановленим репозиторієм. Щоб побачити назву папки треба виконати команду `ls -la` або скорочений варіант `ll`(рис. 3.6).

Дані команди друкують в стандартний вивід в лістинги директорій. Папка називається Docker-Compose-Prometheus-and-Grafana. Після чого необхідно перейти в папку за допомогою команди `cd` – команда командного рядку, яка використовується в системі Linux для зміни поточного розташування робочого каталогу. Команда буде мати вигляд `cd Docker-Compose-Prometheus-and-Grafana`.

Останнім кроком, який потрібно виконати команду `docker-compose up -d` (рис. 3.6), яка з docker-образу створює нові контейнери, а також використовує до них всі конфігурації, що вказані в `docker-compose` файлі. Команда `-d` вказує розгорнути контейнери в фонову режимі, таким чином зберігається доступ до терміналу та мати можливість продовжувати з ним роботу далі.



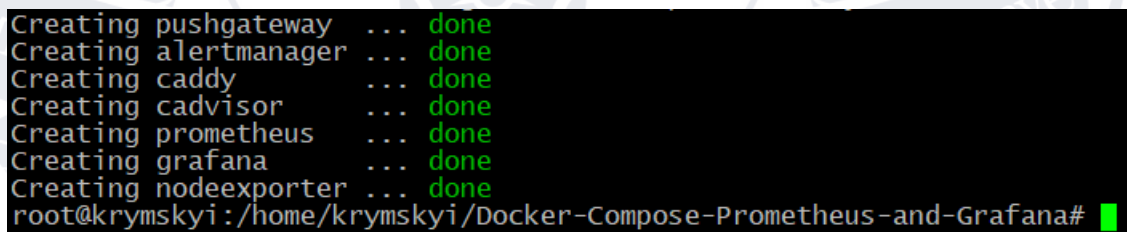
```

root@krymskyi:/home/krymskyi# ll
total 32
drwxr-x--- 5 krymskyi krymskyi 4096 May  8 16:25 ./
drwxr-xr-x 3 root      root      4096 May  7 14:01 ../
-rw-r--r-- 1 krymskyi krymskyi  220 Jan  6 16:23 .bash_logout
-rw-r--r-- 1 krymskyi krymskyi 3771 Jan  6 16:23 .bashrc
drwx----- 2 krymskyi krymskyi 4096 May  7 14:06 .cache/
drwxr-xr-x 9 root      root      4096 May  8 16:25 Docker-Compose-Prometheus-and-Grafana/
-rw-r--r-- 1 krymskyi krymskyi  807 Jan  6 16:23 .profile
drwx----- 2 krymskyi krymskyi 4096 May  7 14:01 .ssh/
-rw-r--r-- 1 krymskyi krymskyi    0 May  8 16:03 .sudo_as_admin_successful
root@krymskyi:/home/krymskyi# cd Docker-Compose-Prometheus-and-Grafana/
root@krymskyi:/home/krymskyi/Docker-Compose-Prometheus-and-Grafana# docker-compose up -d

```

Рисунок 3.6 – розгортання контейнерів системи моніторингу

У результаті має вивестись інформація про те, які контейнери були запуснені (рис. 3.7).



```

Creating pushgateway ... done
Creating alertmanager ... done
Creating caddy ... done
Creating cadvisor ... done
Creating prometheus ... done
Creating grafana ... done
Creating nodeexporter ... done
root@krymskyi:/home/krymskyi/Docker-Compose-Prometheus-and-Grafana#

```

Рисунок 3.7 – успішне розгортання контейнерів

Серед контейнерів системи моніторингу це `pushgateway`, `alert manager`, `caddy` – зворотній проксі сервер із забезпеченням базової аутентифікації для системи моніторингу та підтримки SSL з'єднання, `cadvisor` – збиральник метрик із контейнерів, `prometheus`, `grafana` і `nodeexporter` [33].

На цьому етапі встановлення системи моніторингу завершено, але ще не налаштовано. Щоб переконатись, що контейнери працюють треба виконати команду `docker ps` (рис. 3.8), яка виводить лістинг контейнерів у стандартний вивід терміналу. Для зручності основні контейнери системи моніторингу `Prometheus`, `Grafana`, `node-exporter` були об'єднані в один стек `diploma`.


```

root@krymskiy1:/home/krymskiy1# docker ps | grep diploma
1b732f4202d1   grafana/grafana:8.5.2-ubuntu   "/run.sh"   19 seconds ago   Up 17 seconds   3000/tcp   diploma_grafana.1.i1nonwake4s89k1s7k62
b91ly         65995f8aa673   prom/prometheus:v2.35.0       "/bin/prometheus --c."   24 seconds ago   Up 20 seconds   9090/tcp   diploma_prometheus.1.si5vajwokdefd3nhz
1s52422       5be1fed34a18   prom/node-exporter:v1.3.1     "/bin/node_exporter ..."   About a minute ago   Up 59 seconds   9100/tcp   diploma_node-exporter.1.grr15k11ab710fw
root@krymskiy1:/home/krymskiy1#

```

Рисунок 3.8 – перевірка стану контейнерів

Наступним етапом буде перевірка доступу до сервісів системи моніторингу.

3.3 Доступ до системи моніторингу

Контейнери розгорнуті, система моніторингу встановлена на локальному сервері Linux. Тепер час розібратись як отримати доступ до системи та щось побачити.

У конфігураційному файлі `docker-compose.yml`, який знаходиться у папці з установкою системи моніторингу можна визначити порти, по яких буде здійснений доступ до Prometheus, Grafana та node-exporter.

Сервіси доступні по наступних портах при стандартному налаштуванні портів [34]:

- Prometheus 192.168.0.107:9090 (рис. 3.9)

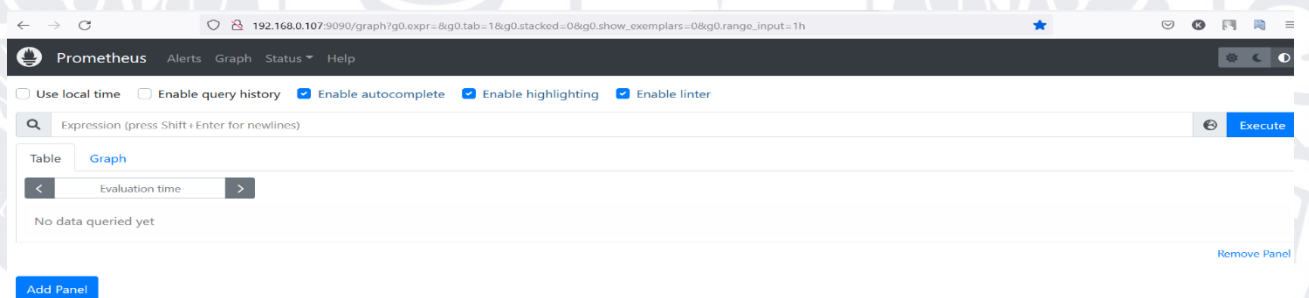


Рисунок 3.9 – доступ та інтерфейс Prometheus

- Grafana 192.168.0.107:3000 (рис. 3.10)

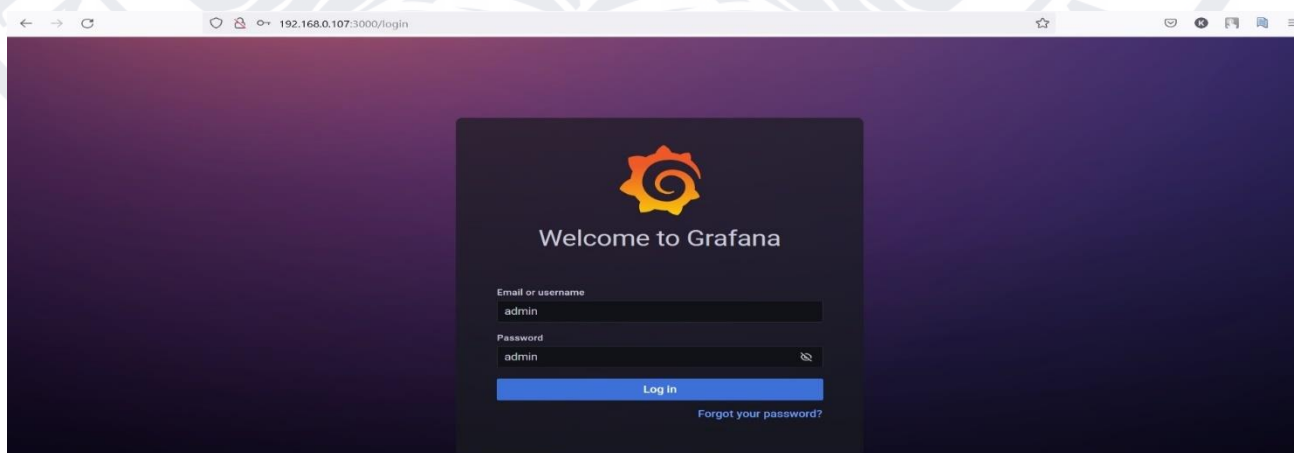


Рисунок 3.10 – доступ та інтерфейс Grafana

Щоб ввійти до node-exporter треба використати дані нижче. Хоча це не дуже потрібно, оскільки через інтерфейс можливо лише перевірити метрики. Точніше сказати подивитись, які саме метрики на сервері збираються за допомогою node-exporter.

- Node Exporter 192.168.0.107:9100 (рис. 3.11)

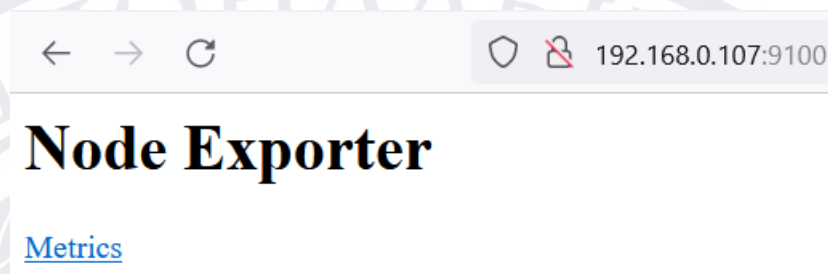


Рисунок 3.11 – доступ та інтерфейс Node-Exporter

За бажання порти доступу можливо змінити на інші у docker-compose.yml файлі порти визначені у вигляді:

ports:

- "3000:3000"

Перше значення це порт, який прослуховує хост, а друге значення це порт самого контейнеру. Тому при бажанні порт, який використовує сервер можна змінити та використовувати не стандартні порти для доступу до системи.

Оскільки це локальний сервер, то доступ здійснюється через IP адресу. Для того, щоб зайти по доменному імені на сервері можна встановити та налаштувати одну з технологій Apache, LiteSpeed, Nginx. Ці технології дозволяють розміщувати дані в глобальній мережі та налаштовувати зону. Для цього потрібно купити, орендувати або зареєструвати домене ім'я та налаштувати зону всі записи CNAME, A , NS тощо відповідно до доменного ім'я [35].

При спробі відкрити адреси зазначені вище буде запрошений пароль та логін для доступу. Пароль та логін були встановлені у розділі 3.1. Для входу в систему потрібно ввести логін admin та пароль admin.

Нюанс при вході в Grafana після вводу логін даних автоматично буде запропоновано змінити пароль, що й потрібно зробити. Важливо використовувати сильні паролі абсолютно повсюди.

Після чого вхід у графічне відображення Grafana буде завершений

3.4 Налаштування системи моніторингу

Оскільки тепер є доступ до системи моніторингу необхідно її налаштувати для збору даних. На даному етапі нічого відображатись крім інтерфейсу не буде. Під налаштування входить підключення Grafana до Prometheus та підключення Prometheus до node-exporter.

За допомогою інтерфейсу Grafana треба перейти в Configuration далі в Data sources та Add data source, де можна підключати різні сервіси такі як Prometheus. Оскільки у даній роботі розробляється система моніторингу на основі Prometheus треба обрати саме Prometheus [36].

У розділі 3.3 були розглянуті адреси по яких можна отримати доступ до Prometheus тому треба ввести адресу 192.168.0.107:9090 важливо виконати правильне налаштування, щоб система працювала(рис. 3.12). Після чого зберегти дані і появиться повідомлення Data source is working. Це означає, що Grafana успішно було підключено до Prometheus.

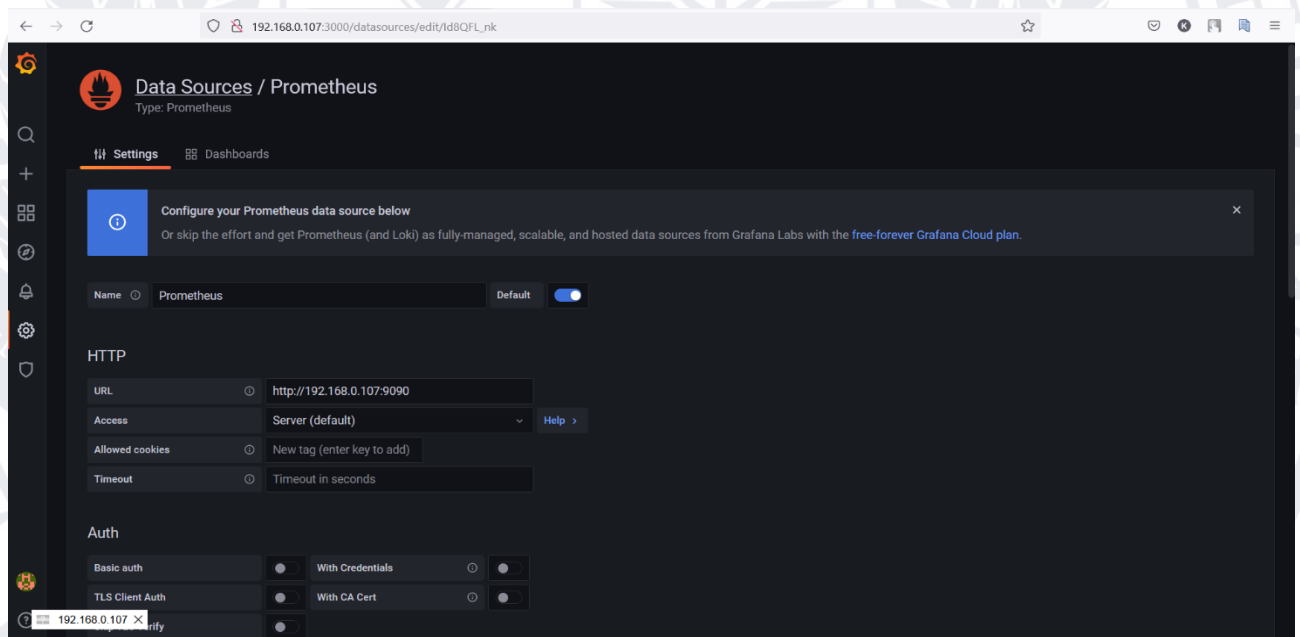


Рисунок 3.12 – налаштування підключення Grafana до Prometheus

Як було зазначено раніше також треба підключити Prometheus до node-exporter. Основний файл конфігурації для Prometheus знаходиться по шляху /var/lib/docker/volumes/diploma_prom-configs/_data/prometheus.yml [37].

У файл необхідно додати Job – роботу або завдання, яке вказує для Prometheus, що потрібно зробити. Тому треба додати досить прості команди конфігурації аби з'єднати Prometheus до node-exporter(рис. 3.13). Важливо вказати вірний порт доступу, який був визначений у розділі 3.3, а саме стандартний порт 9100.


```
- job_name: "node-exporter"

static_configs:
  - targets: ["node-exporter:9100"]
```

Рисунок 3.13 – налаштування конфігурації Prometheus

Файл конфігурації змінився проте Prometheus не оновив ще інформацію, для оновлення потрібно відправити сигнал для Prometheus, щоб контейнер запусився знову. При запуску щоразу автоматично перевіряються файли конфігурації і при перевірці буде виконане з'єднання з node-exporter. За допомогою команди `docker ps | grep diploma_prometheus` треба визначити Id контейнеру та за допомогою команди `docker kill -s SIGHUP 65995f8aa673` відправити сигнал для того, щоб Prometheus оновився. Id контейнера 65995f8aa673 саме тому на цей Id був відправлений сигнал (рис. 3.14).

```
root@krymskyi:/home/krymskyi/grafana-docker-stack# docker ps | grep diploma_prometheus
65995f8aa673   prom/prometheus:v2.35.0   "/bin/prometheus --c..."   2 hours ago   Up 2 hours           9090/tcp   diploma_prometheus.1.s15vajvwokdefd3nhz15524z2
root@krymskyi:/home/krymskyi/grafana-docker-stack# docker kill -s SIGHUP 65995f8aa673
65995f8aa673
```

Рисунок 3.14 – відправлення сигналу для Prometheus

Відтепер якщо перейти до Prometheus по адресі 192.168.0.107:9090 та відкрити меню Status і підрозділ Targets з'явиться node-exporter з'єднаний з Prometheus [38]. Під час успішного з'єднання буде відображатись стан "UP" та будуть відсутні помилки (рис. 3.15).

The screenshot shows the Prometheus web interface at 192.168.0.107:9090/targets. The 'Targets' section is active, showing a list of targets. Two targets are listed, both with a state of 'UP'.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://node-exporter:9100/metrics	UP	instance="node-exporter:9100" job="node-exporter"	8.500s ago	70.719ms	
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	15.334s ago	10.859ms	

Рисунок 3.15 – Prometheus підключений до node-exporter

Відтепер система моніторингу працює і на сервері збираються метрики, які згодом збираються в Prometheus та будуть відображатись у Grafana. Проте Grafana ще буде налаштована у одному з наступних розділів.

3.5 Налаштування візуального відображення у Grafana

Дашборд – набір окремих панелей, розміщених у сітці з набором змінних (наприклад: ім'я серверу, програми та метрики). Змінюючи змінні, можна перемикає дані на дашборді (наприклад: дані з різних метрик CPU, Memory Usage тощо) [39]. Дашборд можна налаштувати, розбивати на секції та фрагментувати представлені в ньому дані відповідно до потреб користувача. У проєкті Grafana велика спільнота і постійно розробляються готові до використання дашборди. Оптимальним рішенням є використовувати готові рішення та налаштовувати відображення під власні потреби та метрики, які збираються у моніторингу.

Серед сотень готових дашбордів найкраще обрати офіційний дашборд від Grafana для node exporter. Цей вибір пояснюється тим, що це один із найбільш продуманих та універсальних дашбордів для відображення абсолютно всіх метрик. Тому достатньо буде видалити не потрібні панелі та метрики з дашборду, щоб залишити лише метрики апаратного забезпечення.

Панель – є базовим елементом візуалізації вибраних показників. У Grafana широкий вибір інструментів з графіками, таблицями, одиничними статусами метрик, тепловими картами. Можна налаштувати стилі та формати відображення кожної панелі, які відображаються. Вільно переміщувати важливі показники вгору дашборду, змінювати розміри, налаштовувати кольори фону, та відображення коли показники в нормі та коли показники досягають критичних показників [40].

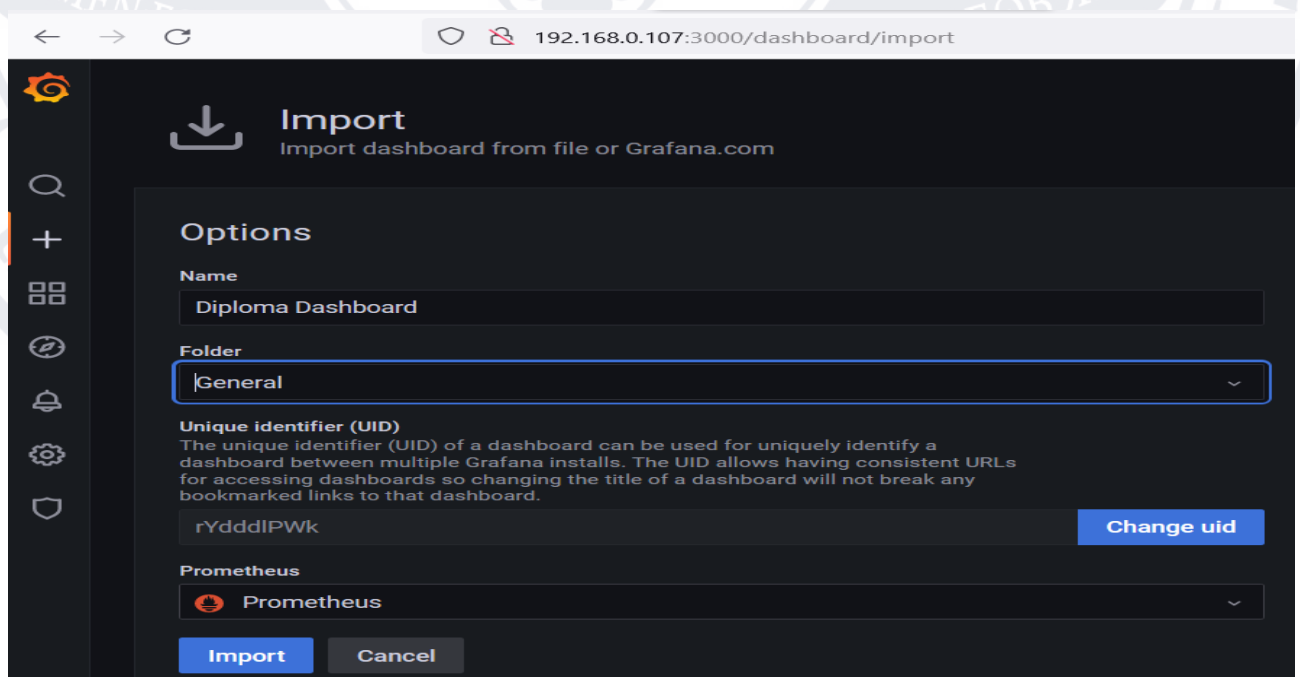


Рисунок 3.16 – Імпортування дашборду grafana

Серед розділів виокремлені:

- Загальні ЦПУ, оперативна пам'ять та дисковий простір

Швидка інформація із 9 панелей про загальний стан сервера. До складу секції входить: загальна загрузка центрального процесора. Два показники +system load – середня загрузка сервера за останні 5 та 15 хвилин, показники розділенні для відстеження ситуації і порівняння як змінилась середня загрузка за 10 хвилин, іноді виконуються короткотривалі важкі процеси, які можуть загрузити систему на кілька хвилин, проте якщо подивитись у зрізі часу – нічого критичного.

- Основні ЦПУ, Оперативна пам'ять та дисковий простір

Всі ці дані також є в загальній секції, але різниця в тому, що в даній секції всього 3 панелі, але з повною інформацією про навантаження центрального процесора, показники оперативної пам'яті та використання дискового простору.

Дані ЦПУ відображають інформація про навантаження системою, користувачами, відсоток часу протягом якого центральний процесор простоював або протягом якого часу система мала не виконані запити, переривання.

Серед даних оперативної пам'яті відображаються: загальна кількість, кількість використаної на даний момент часу, кількість зайнята кешом та буфером обміну, кількість вільної оперативної пам'яті та кількість використаної SWAP пам'яті.

- ЦПУ, Оперативна пам'ять та дисковий простір

Детальна інформація майже по кожному пункту з секції Основні ЦПУ, Оперативна пам'ять та дисковий простір. Складається з 6 панелей та відображає 22 різних показника.

- Дані оперативної пам'яті

Повна інформація про оперативну пам'ять, де можна побачити, детальне використання пам'яті різними процесами системи та виявити потенційне джерело можливої проблеми, що значно полегшить вирішення проблеми.

- Оперативна пам'ять Vmstat

Інформації про помилки оперативної пам'яті кількість сторінок, які були завантажені в систему та з системи, а також кількість помилок, що виникали. Складається з 2 панелей та відображає 5 показників.

- Системні MISC

Інформація про ресурси, що використовуються для роботи самої лише системи. Складається з 9 панелей та відображає 18 показників.

- Дисковий простір

Детальна інформацію про роботу жорсткого диску. Використаний час для запису/ зчитування з диску, швидкість запису/ зчитування з диску, середній сам очікування від диску тощо. Всього 6 панелей із 11 показниками.

- Файлова система зберігання

Інформація про іноди (простіше кажучи кількість файлів) у системі.

3.6 Alert manager. Сповіщення в telegram bot

Alert Manager – це додаток, яка обробляє повідомлення(сповіщення), які надсилаються від Prometheus сервера (наприклад, коли якась із метрик досягла 100% завантаження) і сповіщає користувача за допомогою налаштованого каналу зв'язку [41].

Alert Manager у Grafana в своєму арсеналі має досить великий набір каналів для зв'язку: slack, mail, telegram, discord, MS Teams тощо. Без умовно в кожного каналу зв'язку є свої переваги та недоліки, труднощі в налаштуванні та моменти, які спрощують налаштування. Проте у сучасному світі важко знайти в кого немає додатку Telegram, практично вся молодь користується ним.

Telegram (Telegram) – це багатоплатформовий клауд-месенджер для телефонів, ПК, планшетів, який дозволяє обмінюватись текстовими, аудіо, відео повідомленнями та багато чим іншим [42]. Суттєва перевага Telegram – це підтримка ботів та легкість їхнього налаштування. Саме ботові будуть приходити сповіщення (попередження) від alert manager про критичні показники апаратного забезпечення сервера.

Ще кілька років тому для роботи з alert manager потрібно було б окремо встановлювати контейнер, потім з'єднувати з Grafana в ручну, як це робилось в розділі 3.4 з Prometheus та Grafana. На даний момент часу alert manager уже інтегрований у Grafana і за допомогою стандартного інтерфейсу можна виконувати налаштування сповіщень, каналів зв'язку, створювати правила-тригери для показників метрик(коли показник досягає умови правила -

генерується сповіщення), а також звісно сполучати канали зв'язку з alert manager.

3.6.1 Створення телеграм бота

Перед налаштуванням менеджера сповіщень потрібно підготувати все в телеграмі, а саме створити бота та API token(ключ) та дізнатись особистий chat id телеграму за допомогою, яких і буде здійснено з'єднання між телеграм ботом і менеджером сповіщень.

Поглиблюватись у специфіку та тонкощі створення телеграм ботів не є умовою поставленої задачі, тому бот буде створений за допомогою офіційного боту “BotFather” від розробників телеграму [43].

BotFather – це самий простий спосіб для реєстрації, налаштування й управління іншими телеграм ботами. Робота з ним досить проста та не потребує якихось специфічних навичок від користувача.

За допомогою пошуку в телеграм треба знайти BotFather та запустити його за допомогою кнопки інтерфейсу телеграму. Далі за допомогою текстового повідомлення /newbot можна створити новий власний бот. Буде запропоновано вибрати ім'я бота після обрання імені потрібно придумати псевдонім або username для бота(рис. 3.18).

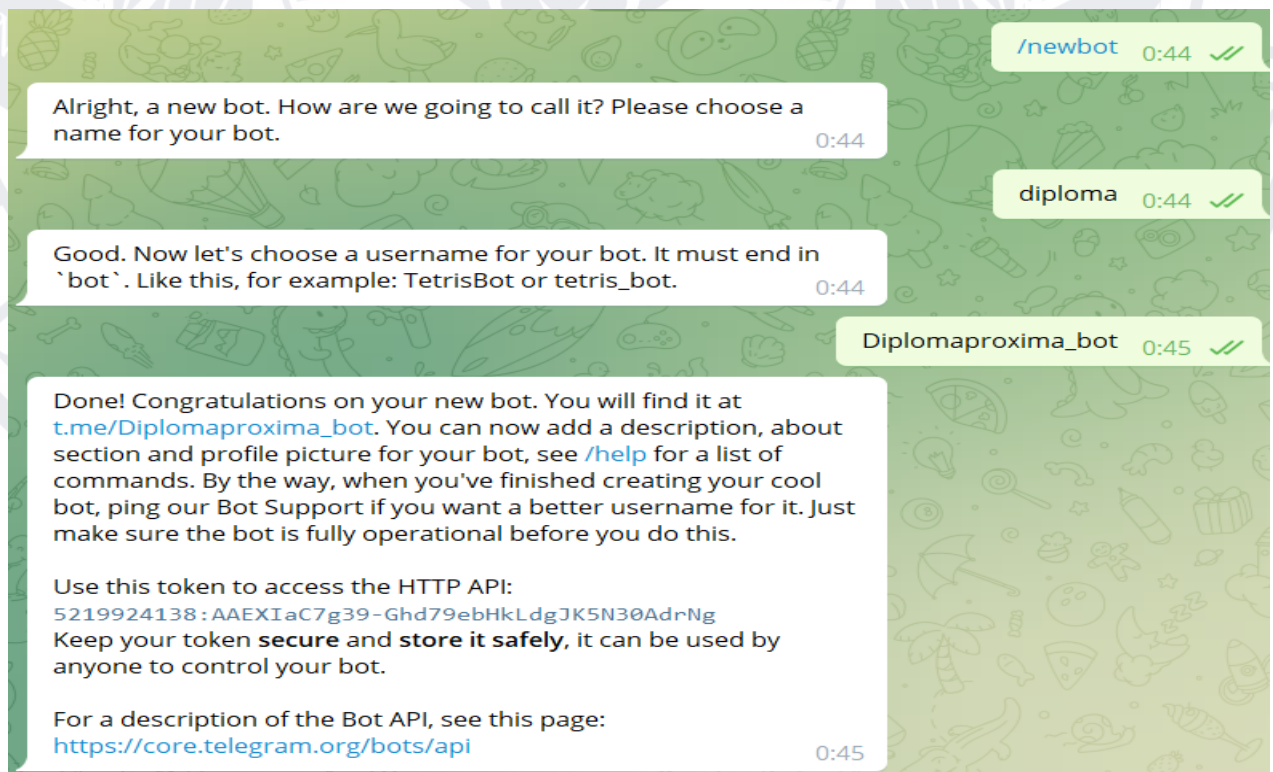


Рисунок 3.18 – реєстрація телеграм бота

Звідси потрібно буде скопіювати та далі використати API ключ.

Як зазначалось вище ще потрібен особистий chat id, у цілях безпеки id показано не буде, але його легко дізнатись за допомогою іншого телеграм бота – IDBot(рис. 3.19)

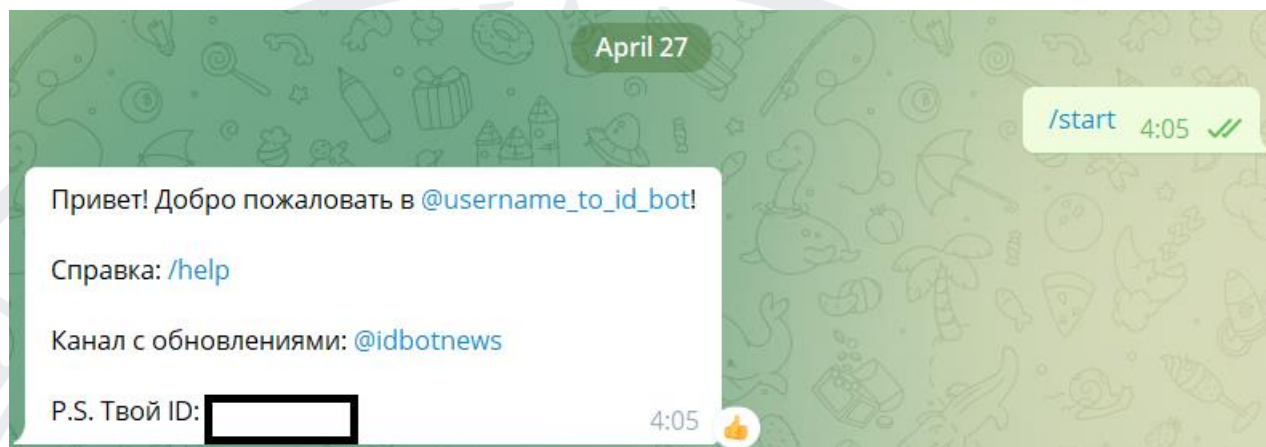


Рисунок 3.19 – особистий id телеграму

Тепер можна запустити створений раніше телеграм бот із псевдонімом Diplomarproxima_bot, при створенні BotFather надіслав посилання на бот t.me/Diplomarproxima_bot потрібно перейти по ньому та активувати бота за допомогою команди /start.

Досить легко за кілька хвилин був створений телеграм бот у який будуть приходити сповіщення від менеджера сповіщень та скопійовані всі необхідні дані для з'єднання менеджера сповіщень та телеграм бота.

3.6.2 З'єднання менеджера сповіщень з телеграм ботом та створення сповіщень.

Оскільки все було підготовлено тепер можна досить легко підключити надсилення сповіщень в телеграм бот. Варто відмітити, що в цілому будь-який канал зв'язку підключається по аналогії, звісно скрізь свої нюанси, але схема подібна.

За допомогою інтерфейсу Grafana треба перейти в розділ сповіщень (Alerting) та вибрати пункт точки сповіщень (contact points). Далі за допомогою кнопок інтерфейсу натиснути додати нову точку сповіщення. У вікні, що появиться з випадаючого списку обрати телеграм і форма заповнення даних зміниться.

Обов'язково треба обрати ім'я для точки сповіщення, яке буде легко розпізнати, оскільки таких точок може бути значна кількість. Зазвичай для кожного типу сповіщення налаштовують різні точки сповіщень. Потрібно

ввести API токен або ключ. Де взяти цей ключ було розглянуто в попередньому розділі. Наступним пунктом ввести особистий ID телеграму. При бажанні також можна налаштувати додаткові опції наприклад повідомлення, яке буде в шапці сповіщення або вимкнути статус окей, який повідомляє, що проблема з метрикою більше не актуальна. Хоча вимикати статус окей не рекомендується адже це важлива інформація.

Конфігурація була завершена і можна відправити тестове повідомлення за допомогою кнопки інтерфейсу тест. Повідомлення успішно прийшло в телеграм бот, що означає все вірно налаштовано та можна зберігати точку сповіщення.

3.6.3 Створення правил-тригерів для сповіщень

Те що сповіщення підключені до телеграм бота звісно добре, але поки що ніяких сповіщень приходить не буде в разі якихось проблем на сервері. Для того, щоб сповіщення почали приходити потрібно створити спеціальні правила або ж критичні точки при досягненні яких буде генеруватись повідомлення про проблему та надсилатись в телеграм.

Правила мають досить гнучку систему налаштувань, але серед всього різноманіття варто виділити операції, яких існує всього чотири: математичні, зменшення, повторна вибірка та загальні значення. Найбільш популярні, які підходять в більшості випадків це загальні значення та математичні опції [44].

У даній задачі будуть використані загальні значення оскільки система моніторингу збирає та відображає дані метрики апаратного забезпечення і це зазвичай числа з даною опцією найзручніше працювати.

В цілому форма для кожного правила стандартна це значення коли показник (більше, менше, середнє, максимальне тощо) показника X протягом Y часу періоду буде створене сповіщення про проблему. Таких правил можна додавати багато, але хорошою практикою є одне правило на один показник. Звісно завжди бувають винятки, але для легкого редагування правил та зрозумілості краще не створювати більше одного правила для одного показника.

Також важливо додати умову при якій менеджер сповіщень буде надсилати повідомлення, що проблема зникла та прийде статус окей. Час потрібно встановлювати індивідуально відповідно до показника наприклад оцінювати значення показника протягом 5 хвилин кожную хвилину, тобто перевірок буде всього 5 і якщо всі з них зі статусом окей – менеджер сповіщень надішле сповіщення, що показник в нормі, тобто прийде статус окей.

В цілому для кожного показника, який моніториться в дашборді потрібно створювати такі правила-тригери, оскільки показників метрик досить багато і переглядати кожну займає певний час і не дуже зручно. Набагато зручніше коли все автоматизовано і в разі проблем прийде сповіщення, а на екрані бачити загальну інформацію про показники серверу.

Є два шляхи для створення тригерів, перший вибираючи якусь конкретну метрику на дашборді та заходити в меню редагування де буде розділ сповіщень. Інший варіант відкрити менеджер сповіщень за допомогою кнопки інтерфейсу зліва та перейти в розділ сповіщень. Проте перший варіант є більш зручним оскільки одразу будуть потрібні метрики для створення сповіщень.

Налаштувавши правило-тригер при проблемі в телеграм бот буде приходити сповіщення з загальним описом проблеми, коротким заголовком, з корисними посиланнями(рис. 3.20).

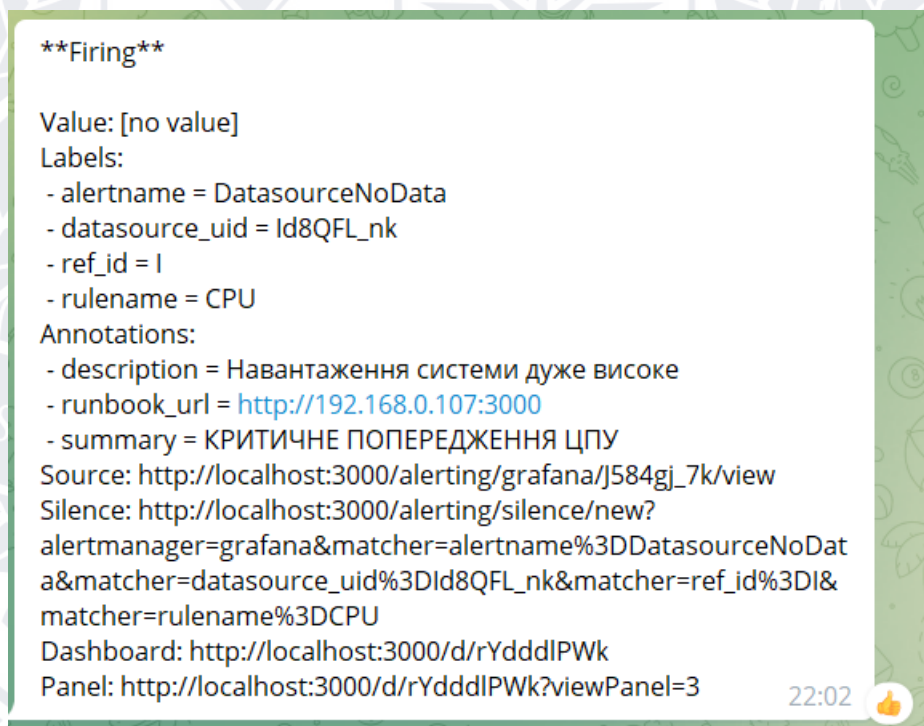


Рисунок 3.20 – приклад сповіщення в телеграм бот

3.7 Висновки до розділу 3

У даному розділі детально розглянутий принцип роботи моніторингової системи, що базується на основі трьох компонентів Prometheus, Grafana та node-exporter. Пояснений процес встановлення моніторингової системи на Linux

сервер, наведені всі потрібні команди як у вигляді тексту так і на рисунках. Пояснені способи доступу до моніторингової системи та механізми налаштування доступів. Моніторингова система була повністю налаштована та підключена до Linux серверу. Створений дашборд для відображення стану апаратного забезпечення на Linux сервері. У кінці кінців створені сповіщення у телеграм бот від системи моніторингу про критичні показники апаратного забезпечення.



ВИСНОВКИ

У даній бакалаврській роботі були визначені основні поняття системи моніторингу апаратного забезпечення. Пояснені механізми роботи моніторингу та розкрита сутність апаратного забезпечення, а також встановлена та налаштована система моніторингу.

Розглянуті та проаналізовані в загальних рисах сучасні системи моніторингу. Визначені основні переваги кожної із розглянутих систем та визначені основні задачі на які націлена та чи інша система моніторингу. На основі проведеного аналізу була обрана платформа на базі якої далі була розроблена та налаштована система моніторингу апаратного забезпечення на Linux сервері.

Пояснений процес встановлення та налаштування Linux сервера на якому далі була встановлена та налаштована система моніторингу на основі Prometheus, Grafana та node exporter.

У результаті можна відслідковувати як основні так і специфічні метрики апаратного забезпечення. Збирається та відображається детально інформація про процесор, оперативну пам'ять і дисковий простір. У разі виникнення проблем із сервером система моніторингу відслідковує метрики та надсилає сповіщення про проблему в телеграм бот.

Створена система моніторингу може бути легко інтегрована на будь-який сервер на основі Linux та відслідковувати інформацію про сервер.

Поставлена задача була виконана в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кенін А. Практическое руководство системного администратора, 2009 год, DHV-СПб. 32с.
2. Amiya Ranjan Mohanty. Machinery Condition Monitoring: Principles and practices, 1st edition, 18 December 2014. 45с.
3. Апаратне забезпечення. URL: https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення 23.03.2022).
4. Gerardus Blokdyk. Server Room Environment Monitoring System a Complete Guide, 2020 Edition (Kindle edition).
5. Brian Brazil. Prometheus: Up & Running: Infrastructure and Application Performance, 1st Edition, 2018, O'Reilly Media, Inc. 58с.
6. Anuran Srivastava, Bahaaldine Azarmi. Learning Kibana 7: Build powerful Elastic dashboards with Kibana's data visualization capabilities, 2nd edition, 2020. 73с.
7. Andrea Dalle Vacche. Mastering Zabbix – Second edition: Learn how to monitor your large IT environments using Zabbix with this one-stop, comprehensive guide to the Zabbix world, 2nd Edition, 2015. 56с.
8. Thomas Kurian Theakanath. Datadog Cloud Monitoring Quick Start Guide, June 2021. 47с.
9. David Carasso. Exploring Splunk, 2012, New York. 15с.
10. Моніторинг клієнтського досвіду. URL: <https://newrelic.com/resources/datasheets/customer-experience-monitoring> (дата звернення 01.04.2022).
11. Wojciech Kocjan, Piotr Beltowski. Learning Nagios – Third Edition, 2016, Packt Publishing. 38с.
12. Daren Hoch. Linux system and Performance monitoring, 2014. 16с.
13. Monitoring dashboard. URL: <https://dashthis.com/monitoring-dashboard/> (дата звернення 03.04.2022).
14. Joel Bastos, Pedro Araujo. Hands-on Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers, May 2019, Packt. 53с.
15. GNU/Linux. URL: <https://uk.wikipedia.org/wiki/Linux> (дата звернення 05.04.2022).

16. Поняття операційна система та її складові. URL: <https://sites.google.com/site/sunlight3555/ponatta-operacijnoie-sistemi-ta-ieie-skladovi> (дата звернення 07.04.2022)
17. Що таке сервер. URL: <https://tebapit.com/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80/> (дата звернення 09.04.2022).
18. Raphael Hertzog, Roland Mas. The Debian Administrator's Handbook. 2003. 71с.
19. Кращі хмарні хостинги 2022. URL: <https://www.tomsguide.com/buying-guide/best-cloud-hosting-services> (дата звернення: 10.04.2022).
20. Що таке SSH. URL: <https://sites.google.com/site/informofinternet/home/protokols/ssh> (дата звернення: 11.04.2022).
21. Mike Lucas. SSH Mastery: OpenSSH, PuTTY, Tunnels, and Keys. 2018. 166с.
22. Що таке VirtualBox і як ним користуватися. URL: <https://geekkies.in.ua/crossplatform/chto-takoe-virtualbox-i-kak-ej-polzovatsja.html> (дата звернення: 20.04.2022).
23. Завантаження Debian iso. URL: <https://www.debian.org/CD/http-ftp/> (дата звернення: 21.04.2022).
24. Полное руководство по SSH в Linux и Windows. URL: <https://hackware.ru/?p=10059> (дата звернення: 23.04.2022).
25. Prometheus docker hub image. URL: <https://hub.docker.com/r/prom/prometheus> (дата звернення: 27.04.2022).
26. Adrian Mouat. Using Docker: Developing and Deploying Software with Containers, 1st Edition, 2016, O'Reilly Media, Inc. 67с.
27. James Turnbull. Monitoring with Prometheus. 2021. 38с.
28. Прометей использует PushGateway для отчетности и сбора данных. URL: <https://russianblogs.com/article/8660485437/> (дата звернення: 27.04.2022).
29. Що таке Docker-compose. URL: <https://dker.ru/docs/docker-compose/overview-of-docker-compose/> (дата звернення: 28.04.2022).
30. Олифер В.Г., Олифер Н.А. Безопасность компьютерных сетей. Горячая линия – Телеком. 109с.
31. Олександр Мізюк. Путівник по Linux. 2021, Київ. 5.2.5 Підключення репозиторіїв.
32. Що таке GitHub і як з ним працювати. URL: <https://training.qatestlab.com/blog/technical-articles/what-is-github-and-how-to-work/> (дата звернення: 29.04.2022).

33. Brian Brazil. Prometheus: Up & Running: Infrastructure and Application Performance, 1st Edition, 2018, O'Reilly Media, Inc. 162с.
34. З'єднання Grafana і Prometheus.
<https://www.metricfire.com/blog/connecting-prometheus-and-grafana/> (дата звернення: 01.05.2022).
35. Cricket Liu, Paul Albitz. DNS and BIND, 5th Edition, 2006, O'Reilly Media, Inc. 74с.
36. Brian Brazil. Prometheus: Up & Running: Infrastructure and Application Performance, 1st Edition, 2018, O'Reilly Media, Inc. 126с.
37. Конфігурація Prometheus. URL:
<https://prometheus.io/docs/prometheus/latest/configuration/configuration/> (дата звернення: 03.05.2022).
38. Brian Brazil. Prometheus: Up & Running: Infrastructure and Application Performance, 1st Edition, 2018, O'Reilly Media, Inc. Chapter 7.
39. Dashboards Grafana. URL: <https://grafana.com/grafana/dashboards/> (дата звернення: 04.05.2022).
40. About Grafana panels. URL: <https://grafana.com/docs/grafana/latest/panels/> (дата звернення: 04.05.2022).
41. AlertManager. URL: <https://prometheus.io/docs/alerting/latest/alertmanager/> (дата звернення: 05.05.2022).
42. Телеграм. URL:
<https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%B3%D1%80%D0%B0%D0%BC> (дата звернення: 07.05.2022).
43. Создание бота в Telegram с помощью BotFather. URL:
<https://lessondelivery.com/telegram/botfather.html> (дата звернення: 09.05.2022).
44. Create alerts Grafana. URL:
<https://grafana.com/docs/grafana/latest/alerting/old-alerting/create-alerts/> (дата звернення: 10.05.2022).

ДОДАТОК А.

Лістинг файлу docker-compose.yml

```
version: "3.9"
```

```
services:
```

```
  grafana:
```

```
    image: grafana/grafana:8.5.0-ubuntu
```

```
    ports:
```

```
      - "3000:3000"
```

```
    volumes:
```

```
      - grafana-data:/var/lib/grafana
```

```
      - grafana-configs:/etc/grafana
```

```
  prometheus:
```

```
    image: prom/prometheus:v2.35.0
```

```
    ports:
```

```
      - "9090:9090"
```

```
    volumes:
```

```
      - prom-data:/prometheus
```

```
      - prom-configs:/etc/prometheus
```

```
  node-exporter:
```

```
    image: prom/node-exporter:v1.3.1
```

```
    ports:
```

```
      - "9100:9100"
```

```
    volumes:
```

```
      - /proc:/host/proc:ro
```

```
      - /sys:/host/sys:ro
```


- /:/rootfs:ro

command:

- '--path.procfs=/host/proc'

- '--path.sysfs=/host/sys'

- '--collector.filesystem.mount-points-exclude'

-

'^/(sys|proc|dev|host|etc|rootfs/var/lib/docker/containers|rootfs/var/lib/docker/overlay2|rootfs/run/docker/netns|rootfs/var/lib/docker/aufs)(\$\$|/)'

volumes:

grafana-data:

grafana-configs:

prom-data:

prom-configs:

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Кримський Руслан Олегович

Прізвище, ім'я, по батькові

Факультет інформаційних і прикладних технологій

Факультет

122 Комп'ютерні науки

Шифр і назва спеціальності

Сучасні інформаційні технології та програмування

Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка системи моніторингу апаратного забезпечення на Linux сервері» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

дата

підпис