МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)
_____червня 2024 р.____

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - «Комп'ютерні науки»,
освітньо-професійної програми «Інформатика»
на тему: «Система діагностики стану серверної інфраструктури на основі мережевого моніторинга»
здобувача групи КНдн-01с Черніявського Віктора Анатолійовича

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

_____ Віктор ЧЕРНІЯВСЬКИЙ
(підпис)

Керівник, Владислав П'ЯТАЧЕНКО
асистент,
доктор філософії

_____
(підпис)

Суми – 2024

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 - «Комп'ютерні науки», освітньо-професійної програми «Інформатика»
здобувача групи КНдн-01с Черніявського Віктора Анатолійовича

1. Тема роботи: «Система діагностики стану серверної інфраструктури на основі мережевого моніторинга»
затверджую наказом по СумДУ від *«26» квітня 2024 р. № 0438-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 10 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналітичний огляд. 2) Вибір методу рішення. 3) Проектування клієнт-серверної системи. 4)Розробка інформаційної системи моніторингу серверної інфраструктури. 5) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *концептуальна схема системи та мережевого обладнання, презентаційні результати роботи (20 слайдів).*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналітичний огляд.*	*15.05.24-19.05.24*	
2	*Вибір методу рішення*	*20.05.24-21.05.24*	
3	*Проектування клієнт-серверної системи.*	*22.05.24-25.05.24*	
4	*Розробка інформаційної системи моніторингу серверної інфраструктури*	*26.05.24-01.06.24*	
5	*Аналіз отриманих результатів*	*02.06.24-03.06.24*	
6	*Оформлення пояснювальної записки до кваліфікаційної роботи*	*04.06.24-06.06.24*	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 44 стр., 19 рис., 1 додаток, 21 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі забезпечення якісної та стабільної роботи серверної інфраструктури шляхом діагностування збоїв в роботі мережевого програмного забезпечення та обладнання і реалізації методів швидкого реагування на відхилення від штатного режиму функціонування.

Об’єкт дослідження — процес моніторингу серверної інфраструктури.

Мета роботи — установка та налаштування моніторингу серверної інфраструктури за допомогою інструментів моніторингу..

Методи дослідження — формування порівняльної таблиці інструментів моніторингу, моніторинг мережі за допомогою обраного інструменту.

Результати — проаналізовано інструменти мережевого моніторингу. Розроблено систему моніторингу мережевої інфраструктури, яку можна масштабувати та використовувати для корпоративних мереж. Реалізовано механізми моніторингу доступності серверу, вхідного трафіку, використання внутрішніх ресурсів системи, пам'яті та процесора.

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, АРХІТЕКТУРА КЛІЄНТ-СЕРВЕР, МОНІТОРИНГ, СИСТЕМА МОНІТОРИНГУ ZABBIX, ZABBIX AGENT.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AWS – Amazon Web Services;

CPU – Central processing unit;

EC2 – Elastic Compute Cloud.

ВСТУП

Робота присвячена розв’язанню важливої практичної задачі забезпечення якісної та стабільної роботи серверної інфраструктури шляхом діагностування збоїв в роботі мережевого програмного забезпечення та обладнання і реалізації методів швидкого реагування на відхилення від штатного режиму функціонування. Задача моніторингу серверного устаткування та процесів, що відбуваються в рамах клієнт-серверної взаємодії є актуальною, оскільки сучасне інтернет середовище характеризується ростом числа підключень до мережі користувацьких пристроїв різних типів, що зумовлює зростання обсягів передаваної інформації мережею та відповідно навантаження на серверну сторону мережевої системи.

Процеси діджиталізації вимагають від сучасного бізнесу функціональної та ефективної серверної інфраструктури. Так припинення роботи сервера може призвести до втрати продуктивності, прибутку та навіть клієнтів. Моніторинг дозволяє вчасно виявляти проблеми та уникати відмов. Діагностування проблеми шляхом інспекцій та моніторингу також допомагає виявляти аномальну активність, що може свідчити про спроби злому або кібератаки. Вчасна реакція на такі події дозволяє запобігти або мінімізувати збитки. Моніторинг допомагає визначити, які ресурси сервера використовуються неефективно або надмірно, що дозволяє оптимізувати розподіл ресурсів та зменшити витрати на обслуговування. Аналіз даних моніторингу може допомогти вдосконалити робочі процеси та визначити найефективніші шляхи використання ресурсів серверів.

З вище наведеного виходить, що тема дослідження "система діагностики стану серверної інфраструктури на основі мережевого моніторинга" є актуальною та затребуваною. Об’єктом дослідження є процес моніторингу серверної інфраструктури. Предметом – метрики функціонування мережевої серверної системи.

В системах серверної інфраструктури якісна система моніторингу метрик

системи та вчасного реагування на відхилення від штатної роботи дозволить спростити процеси адміністрування та, як наслідок, вплине на позитивний користувацький досвід взаємодії з системою, що збільшить популярність сервісу у користувачів.

Моніторинг системи є комплексною задачею, яка вимагає як набору інструментів для діагностики обраних вузлів так і проектування архітектури серверної інфраструктури і інтеграції інструментів моніторингу і реагування в неї. В рамках роботи розглянуто створення єдиної платформи, що забезпечує комплексний підхід до моніторингу метрик IT-інфраструктури з автоматизованими сценаріями реагування та поєднанням можливостей ряду сучасних інструментів моніторингу.

Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису програмного забезпечення системи моніторингу, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ СИСТЕМ

1.1 Загальні характеристики системи моніторингу мережевого обладнання

В рамках клієнт-серверної взаємодії забезпечення якісної, безперебійної роботи є важливою складовою підтримки програмного рішення. Важливими характеристиками для користувача, який виконує роль клієнта в процесі мережевої взаємодії, є швидкість відповіді сервера та надійність і стабільність з'єднання. Ці аспекти описують важливість того, щоб мережеве обладнання реагувало на запити від клієнтів швидко і без затримок, без втрат пакетів або перерв у з'єднанні.

Втім клієнт-серверна система є значно складнішою з точки зору архітектури, ніж це подається з клієнтського боку. В ролі сервера може постати складна система з розподіленими обов'язками та обчислювальними потужностями. Окрім розмежування програмного продукту на базу даних та веб-сервер, мережева архітектура серверної частини проекту може включати в себе системи кешування для покращення продуктивності, системи безпеки для захисту конфіденційності та цілісності даних, механізми балансування навантаження для розподілу трафіку між серверами та моніторингу роботи системи для виявлення та виправлення проблем. Також можуть використовуватися технології реплікації даних для забезпечення резервування та надійності системи в разі відмови обладнання.

Таким чином важливого значення для серверної частини набувають безпека, тобто можливість забезпечувати захист від вторгнень та зловживань; масштабованість, з метою забезпечити потрібні ресурси та пропускну здатність. Крім того, важливим аспектом для серверної частини є ефективне управління ресурсами, щоб забезпечити оптимальне використання обчислювальних потужностей та мережевих ресурсів.

Логічним наслідком важливості описаних характеристик стає процес моніторингу параметрів серверу, який і є ключовою складовою управління серверною інфраструктурою.

Для забезпечення ефективності та надійності роботи серверів необхідно постійно відстежувати різноманітні параметри. Деякі з найважливіших параметрів, які слід відслідковувати в процесі роботи системи, включають:

1. Використання ресурсів: Моніторинг центрального процесора (CPU), обсягу оперативної пам'яті (RAM) та дискового простору дозволяє вчасно виявляти перевантаження або нестачу ресурсів, що може призвести до погіршення продуктивності або навіть відмови системи.

Додатково до моніторингу CPU, RAM та дискового простору, важливо відстежувати температуру обладнання та стан систем охолодження, щоб запобігти перегріву та потенційним відмовам. Нижче наведено приклад наскільки сильно ефективність використання ресурсів впливає на прибутки компанії. Відповідно до звітів Amazon та Google можна стверджувати наступні кореляції[1]: використовуючи моніторинг реальних користувачів, Amazon виявив, що кожні 100 мс затримки коштують їм 1% в продажах; Google, при аналізі часу завантаження пошукового двигуна, різниця в 0.5 с при завантаженні призводила до 20% зниження трафіку пошуку.

Ці приклади показують, як важливо відстежувати ресурси та реагувати на них в реальному часі, щоб забезпечити ефективну роботу системи та максимальний прибуток.

2. Мережевий трафік: Моніторинг обсягу вхідного та вихідного мережевого трафіку допомагає виявити проблеми зі з'єднанням або атаки на сервер. Такий підхід також повинен включати аналіз пакетів для виявлення небажаного трафіку та потенційних зловмисних дій, а також оцінку якості сервісу (QoS) для гарантування відповідності SLA.

Використовуючи AI-Native Networking Platform, компанія Juniper Networks допомогла німецькому рітейлеру електроніки “Expert” оптимізувати управління мережею та користувацький досвід. Завдяки AI-платформі, “Expert” отримав покращені оперативні дані та автоматизацію, що значно підвищило надійність та безпеку мережі [2].

Команда Toyota Motor North America інтегрувала Datadog, AI-двигун компанії, який надає автоматизовані сповіщення, аналізи та визначення кореневих причин на основі даних спостереження з усієї платформи Datadog. Використання Datadog дозволило ефективно скоротити час на виявлення та вирішення проблем[2].

3. Стан служб і процесів: Перевірка доступності та стабільності важливих служб і процесів, таких як веб-сервери, бази даних та інші, є критичною для забезпечення безперебійної роботи системи.

Важливо також впровадити автоматизовані процедури відновлення для служб та процесів, які можуть автоматично перезапускати або відновлювати їх у разі відмови. Моніторинг продуктивності серверів, затримки мережі та відгуків користувачів дозволяє Netflix виявляти та усувати потенційні проблеми, перш ніж користувачі зіткнуться з будь-якими перервами[3].

4. Журнали подій: Моніторинг та аналіз журналів подій допомагає виявити аномальні події, помилки та попередження про можливі проблеми з безпекою або роботою системи.

Інтеграція з системами управління подіями та інформацією про безпеку (SIEM) може значно покращити здатність до аналізу журналів та виявлення складних загроз.

Приклад з реального життя для моніторингу журналів подій та інтеграції з SIEM: у галузі охорони здоров’я моніторинг журналів подій допомагає виявляти незвичайні дії, такі як несанкціонований доступ до медичних даних пацієнтів. Наприклад, виявлення спроб доступу до медичних записів пацієнтів з підозрілих IP-адрес може вказувати на можливий випадок порушення конфіденційностію[4].

5. Доступність: Постійна перевірка доступності серверів та їхніх послуг допомагає вчасно виявляти відмови та забезпечує високу доступність системи для користувачів. Розгортання резервних серверів та використання технологій віртуалізації може забезпечити високу доступність та відмовостійкість системи.

Звернемось ще раз до сфери охорони здоров’я. В цій надважливій галузі функціонуючі системи мають бути доступні 24/7, використовуючи резервні сервери та налагоджуя систему мониторінгу медичні установи убезпечують нагляд за критично важливими елементама загальної мережі. Госпіталі на заході користуються високодоступними серверами для забезпечення сталого доступу до електронних медичних записів паціернтів (EHR). Навіть якшо один з серверів припине свою роботу, медичний завжди буде мати доступ до резервних[5].

Цей підхід до организації робочого процесу забезпечить високу місцність та відмовостійкість будь яких систем.

6. Використання мережевих портів: Моніторинг навантаження на мережеві порти допомагає виявити перевантаження мережі та оптимізувати розподіл трафіку.

Моніторінг сесій та аналіз протоколів допоможе в оптимізувати використання мережевих портів та підвисить загальну продуктивність мережі.

Фінансові установи: В цій сфері швидкість відгуку, безпека та надійність мережі є необхідними для якісної роботи. Кожна система має ліміт для навантаження мережевих портів. Моніторинг виявляє ситуації коли трапляється перевантаження. Системний адміністратор отримавши метріки має можливість завчасно оптимізувати розподіл трафіку.

Звичайною є практика раннього виявлення DDoS-атак, спостерігаючи за навантажнням мережевих портів[6].

Неможливо спроектувати та спрограмувати систему так, щоб вона сто відсотків часу працювала безвідмовно. Але вчасне реагування на виникаючі позаштатні ситуації за допомогою моніторингу є запорукою успіху та задоволених клієнтів.

1.2 Огляд програмних рішень мережевого моніторингу

Завдання діагностики відхилень від штатного режиму роботи в мережі вимагає забезпечити набір інструментів, які здатні відстежувати та контролювати роботу серверів. Залежно від програмного продукту мережева

архітектура може покладатись як на прості та легкі у використанні інструменти, так і комплексні рішення, для реалізації більш розширених функцій та можливостей. Так і різні серверні архітектури можуть вимагати різних підходів до моніторингу. Наприклад, моніторинг фізичних серверів може відрізнятися від моніторингу віртуальних серверів або хмарних сервісів.

Швидкі та постійні зміни в технологічному світі спонукають до появи нових програмних рішень, які можуть відповідати новим вимогам та викликам у сфері моніторингу серверів, це в свою чергу відображається в розширенні набору інструментарію діагностики мережі, який через зміни в технологіях та конкуренцію між компаніями виробниками програмного забезпечення зазнає постійного вдосконалення та внесення нововведень для привертання користувачів.

Prometheus. Це потужне та гнучке відкрите програмне забезпечення для моніторингу систем, яке володіє великою спільнотою користувачів та розширюваністю за допомогою різноманітних плагінів та інтеграцій. Prometheus - це система моніторингу та оповіщень, яка збирає метрики з різних цілей через експортерів та шлюз PushGateway. Має набір інструментів для аналізу даних та підсистему сповіщень. Для візуалізації результатів моніторингу краще використовувати в зв'язці з Grafana (рис.1.1).

Як переваги програмного рішення розглядають[7] активну спільноту, що спрощує підтримку програмного рішення, гнучку архітектуру, що дозволяє легко масштабувати систему та налаштовувати її під конкретні потреби та ефективну систему моніторингу з можливістю візуалізації метрик та генерації сповіщень. Серед недоліків можна розглянути відсутність готових засобів для візуалізації даних, система вимагає більшої кількості ресурсів для масштабування на великі обсяги даних порівняно з деякими іншими інструментами.

Рисунок 1.1 – Інтерфейс системи моніторингу Prometheus з Grafana

Також Prometheus може вимагати додаткових зусиль для налаштування сповіщень, оскільки ця функціональність може бути менш розвиненою порівняно з іншими системами.

Datadog. Це хмарне рішення моніторингу, яке надає широкий набір функцій[8], включаючи моніторинг інфраструктури, додатків та сервісів, а також аналітику та сповіщення про проблеми. Надає широкий спектр функцій та інструментів для збору, візуалізації та аналізу метрик, журналів, трейсів та інших даних про роботу систем. Значною перевагою платформи Datadog є наявність інтерфейсу користувача (рис. 1.2) та простого процесу встановлення та налаштування. Це програмне рішення надає можливість зберігати дані в хмарному середовищі, що забезпечує їх доступність та надійність навіть у випадку відмови локальних серверів. Втім наведені переваги зручності визначаються пропрієтарним способом розповсюдження програмного забезпечення. Відносно високі витрати на користування можуть зробити програмне рішення недоступним для користувачів з обмеженим бюджетом. В порівнянні з альтернативними рішеннями Datadog має обмежені можливості контролю та налаштувань мережі.

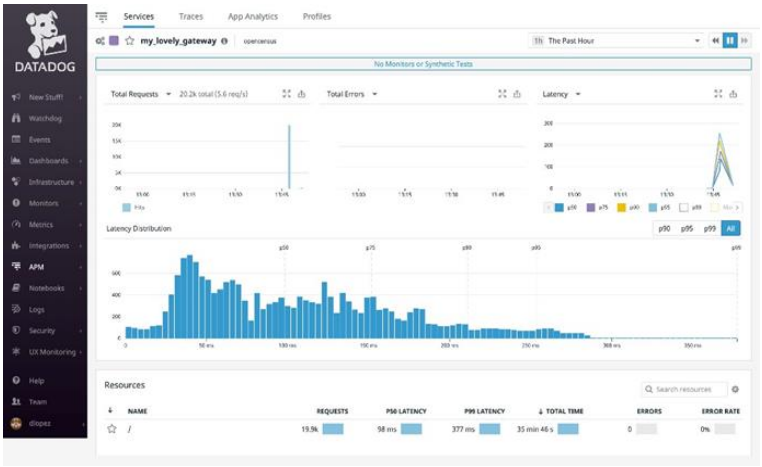


Рисунок 1.2 – Інтерфейс системи моніторингу Datadog

Хмарний підхід в формуванні мережевої архітектури як основну складову проектування системи потребує доступу до інтернету для повного функціонування, що може бути проблематичним для закритих системи, з обмеженнями мережі.

Zabbix, система моніторингу з відкритим кодом, славиться своєю легкістю у встановленні та налаштуванні, а також можливістю розширення за допомогою різноманітних шаблонів та модулів. Вона пропонує функції візуалізації даних, що ґрунтується на збережених інформаційних потоках, що сприяє огляду навантаження на апаратні ресурси. Доступ до всіх звітів та статистики Zabbix, а також параметрів конфігурації, можна отримати через веб-інтерфейс (рис. 1.3). Цей веб-інтерфейс забезпечує можливість отримання інформації про стан мережі та працездатність серверів навіть з віддалених місць. При належному налаштуванні[9] Zabbix стає ключовим елементом моніторингу IT-інфраструктури. Це стосується як невеликих компаній з декількома серверами, так і великих підприємств з обширною IT-інфраструктурою. Zabbix є безкоштовним програмним забезпеченням, розробленим та поширюваним за ліцензією GPL (General Public License версії). Програмне рішення забезпечу автоматичне виявлення серверів і мережевих пристроїв, розподілений моніторинг із централізованим WEB адмініструванням, підтримку механізмів опитування та захоплення серверне програмне забезпечення для Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD.

Рисунок 1.3 – Інтерфейс системи моніторингу Datadog

В системі наявні агенти, які здатні розсилати повідомлення про попередньо визначені події високорівневий огляд ресурсів, що відстежуються ведення журналу аудиту, електронною поштою.

Nagios. Це один з найпоширеніших інструментів моніторингу, який володіє довговічністю та великою кількістю розширень, що дозволяють моніторити різноманітні системи (рис. 1.4).

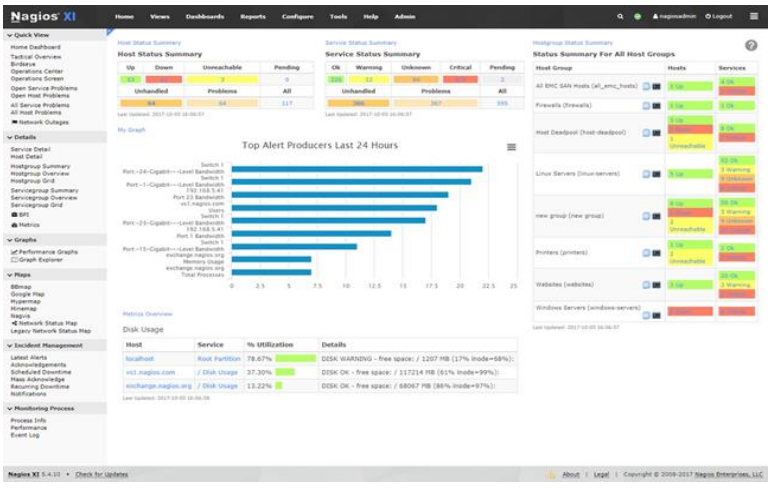


Рисунок 1.4 – Інтерфейс системи моніторингу Nagios

Основна функціональність системи моніторингу Nagios реалізована через ядро Core 4, що забезпечує високий рівень продуктивності за рахунок меншого споживання ресурсів сервера[10]. За допомогою плагінів реалізована можливість

інтегрувати інструмент практично з будь-яким типом стороннього програмного забезпечення. В системі реалізовано централізоване бачення всієї IT-інфраструктури та механізм автоматичного перезапуску додатків у випадку виявлення збоїв у їх роботі. Серед основних принципів програмне рішення розраховане на багатокористувацький доступ до системи, з можливостями обмежень доступу, які дозволяють налаштувати видимість компонентів для користувачів.

Розширювана архітектура Nagios дає можливість за потреби розширити функціонал системи. Варто зазначити важливі складові цієї програми: моніторинг мережевих служб (SMTP, HTTP), моніторинг стану хостів, підтримку віддаленого моніторингу через шифровані канали SSH чи SSL.

Архітектура модулів програмного рішення дозволяє здійснити зв'язку плагінів більшістю мов програмування (Shell, C, Perl, Python, PHP, C#) та має систему відправки сповіщень у разі виникнення проблем із службами або хостом, автоматичну ротацію лог-файлів.

Недоліками ж системи можна вважати показники моніторингу зосереджені на мережевій взаємодії та не рекомендовані до роботи з серверами на операційній системі Windows.

1.3 Хмарне рішення

Диджиталізація більшості сучасних процесів зумовлює тенденцію, що більшість сучасних бізнесових проектів покладаються на певну комп'ютерну систему.

Стабільну популярність демонструють хмарні серверні рішення, оскільки компанії, що тільки починають діяльність, зацікавлені в стабільному, ефективному та не бюджетовимогливому рішенні. Втім і для старіших компаній необхідність підтримання, оновлення та забезпечення сталої якості мережевого обладнання часто стають наріжним каменем, що вимагає шукати альтернативних рішень.

Таким рішенням часто стає, наприклад, платформи хмарних обчислень від Amazon, Microsoft та Google, які направленні на підтримку новими компаніями використання хмарних серверних архітектур, і пропонують міграцію з існуючої архітектури в їхнє хмарне середовище для давно існуючих компаній.

Amazon Web Services (AWS) — це комплексна платформа хмарних обчислень, яку пропонує Amazon.com. AWS надає широкий спектр віддалених обчислювальних служб, включаючи обчислення, зберігання даних, передачу даних по мережі, аналітику, мобільні застосунки та інструменти для розробників. Ці служби працюють у різних географічних регіонах світу і доступні через HTTP, використовуючи архітектурний стиль REST та протокол SOAP1.

AWS дозволяє користувачам мати у своєму розпорядженні віртуальний кластер комп’ютерів, доступний через Інтернет. Віртуальні машини AWS мають більшість показників реального комп’ютера, серед яких апаратні пристрої, операційна система на вибір користувача, мережа та попередньо встановлені прикладні програми.

Платформа AWS спроектована, як поєднання моделі "інфраструктури, як послуги" (IaaS), "платформи, як послуги" (PaaS) і "програмного забезпечення, як послуги" (SaaS). Так, відповідно до заявленого розробником, AWS пропонує гнучкі, надійні, масштабовані, прості у використанні та економічно ефективні рішення хмарних обчислень.

Користувачі можуть вибирати серед різноманітних сервісів AWS, відповідно до своїх потреб, і оплачувати лише за ті ресурси, які вони використовують. Це робить AWS популярним вибором для бізнесів різного розміру, від стартапів до великих корпорацій[11].

Microsoft Azure є однією з провідних хмарних платформ, яка надає широкий спектр сервісів для різних бізнес-потреб. Azure дозволяє користувачам легко створювати, тестувати, розгортати та управляти додатками та сервісами через глобальну мережу центрів обробки даних Microsoft. Ось деякі ключові особливості та сервіси, які пропонує Azure:

1 Обчислювальні Сервіси: Azure Virtual Machines, Azure Kubernetes Service (AKS), та Azure Functions для різних потреб в обчислювальних ресурсах.

2 Сховище Даних: Azure Blob Storage для неструктурованих даних, Azure File Storage для файлових систем, та Azure Queue Storage для зберігання великих обсягів повідомлень.

3 Бази Даних: Azure SQL Database, Azure Cosmos DB, та Azure Database for MySQL для управління структурованими даними.

4 Інтеграція та Аналітика: Azure Logic Apps для автоматизації бізнес-процесів, Azure Stream Analytics для обробки потокових даних, та Azure Data Factory для інтеграції та перетворення даних.

5 Інтелектуальні Сервіси: Azure Cognitive Services для створення додатків з можливостями штучного інтелекту, та Azure Machine Learning для розробки та тренування моделей машинного навчання.

6 Безпека та Ідентифікація: Azure Active Directory для управління ідентифікацією та доступом, та Azure Security Center для забезпечення безпеки хмарних ресурсів.

Azure постійно розвивається, додаючи нові сервіси та можливості, щоб задовольнити зростаючі потреби бізнесу в цифровій трансформації. Використання Azure може допомогти компаніям зменшити витрати на IT-інфраструктуру, підвищити ефективність та гнучкість, та прискорити інноваційні процеси.

Google Cloud Platform (GCP) відома своєю сильною інтеграцією з аналітикою великих даних, машинним навчанням та штучним інтелектом, що робить її популярним вибором для компаній, які займаються інноваціями в цих областях. Ось детальніше про ключові можливості GCP:

1 Обчислювальні Ресурси: GCP пропонує віртуальні машини Google Compute Engine, контейнери в Google Kubernetes Engine, та безсерверні обчислення з Google Cloud Functions.

2 Сховище Даних: GCP надає різноманітні опції для сховища, включаючи Google Cloud Storage для неструктурованих даних та Cloud SQL для реляційних баз даних.

3 Біг Дата та Аналітика: BigQuery для аналізу великих даних, Cloud Dataflow для потокової та пакетної обробки даних, та Cloud Dataprep для підготовки даних для аналізу.

4 Машинне Навчання та Штучний Інтелект: TensorFlow та Cloud Machine Learning Engine для створення та тренування моделей машинного навчання, та API Vision, Speech та Natural Language для інтеграції можливостей штучного інтелекту в додатки.

5 Інтеграція та Розробка: Cloud Endpoints для створення та управління API, та Firebase для розробки мобільних та веб-додатків.

6 Безпека та Управління: Cloud IAM для управління доступом та ідентифікацією, та Cloud Security Command Center для моніторингу та управління безпекою хмарних ресурсів.

GCP також пропонує глибоку інтеграцію з іншими продуктами Google, такими як Google Ads, Google Analytics, та G Suite, що дозволяє користувачам легко використовувати дані та аналітику для підвищення ефективності бізнесу. Крім того, GCP має сильну підтримку відкритого програмного забезпечення та спільноти розробників, що сприяє інноваціям та співпраці.

Праці [12-14] визначають наступні характеристики хмарних платформ (табл. 1.1). Серед спільних характеристик ці публікації визначають широку доступність за географічним положенням, помірну безпеку середовища та корпоративну спрямованість постачальника.

Таблиця 1.1– Хмарні платформи.

Параметр	Amazon Web Service	Microsoft Azure	Google Cloud Platform
Інфраструктура (IaaS)	Колекція апаратних/програмних елементів/інтерфейс для	Amazon - EC2 (Xen	Microsoft common language runtime

Параметр	Amazon Web Service	Microsoft Azure	Google Cloud Platform
	доступу до хмарних обчислювальних послуг/віртуалізованих ресурсів.	virtualization Engine)	(CLR) (Hypervisor-V)
Обчислювальні послуги (PaaS)	Надають IT як послугу через Інтернет або AWS Beanstalk, Amazon Lightsail, AWS Batch, AWS Platform-as-a-service (PaaS), Function-as-a-service (FaaS)	Service Fabric, Azure Batch	Google App Engine; графічний процесор; Docker Container
Мережеві технології	Дозволяють безпосередньо і безпечно доступатися до мережевої інфраструктури. Amazon дозволяє використовувати VPC послуги; Amazon Route 53 послуги, The Cloud Front послугу, та AWS Direct Connection.	Забезпечує кешування філій, прямий доступ до послуг, допомагає у формуванні основного керівництва мережі, DNS послугу та балансування навантаження	Cloud Interconnect, Cloud VPN, Carrier Peering та Direct Peering
Технології зберігання	Підтримка і управління даними та доступ до них за запитом через мережу.	Можуть використовуватись наступні послуги Amazon: прості сховища, Amazon Glacier,	Azure дозволяє масштабоване зберігання об'єктів, зберігання текстових та бінарних даних. Azure Files: управління файлами для розгортання у хмарі на локальних

Параметр	Amazon Web Service	Microsoft Azure	Google Cloud Platform
		Amazon Elastic Block Store, та Amazon Elastic File System.	установках. Azure Queues: обмін повідомленнями
Підтримка реляційних баз даних	Дозволяє користувачам доступатися до бази даних без встановлення програмного забезпечення/налаштування середовища.	Amazon - RDS, SQL, MySQL та Oracle	Azure SQL, Azure-SQL-DW
Послуги резервного копіювання	Відновлення/резервне копіювання даних або програм віддалено або онлайн.	Glacier Archival storage, Recovery backups, Site recovery	Near-line, Cold-line часто використовуються для доступу до даних
Типи моделей послуг	Всі обчислювальні моделі - IaaS, PaaS, SaaS, але основний акцент на IaaS, PaaS	Всі три обчислювальні моделі IaaS, PaaS, SaaS, але основний акцент на SaaS	Всі три обчислювальні моделі IaaS, PaaS, SaaS, але основний акцент на PaaS
Фокус	Пропозиції, орієнтовані на підприємства та публічна хмара (поза приміщенням)	Інтеграція з інструментами Microsoft, відкритий код та гібридна хмара (на/поза приміщенням)	Відкритий код та портативність, гібридна хмара (поза приміщенням)

Параметр	Amazon Web Service	Microsoft Azure	Google Cloud Platform
Характер послуг	Пропонує послуги, орієнтовані на підприємства	Не повністю готовий для підприємств	Спроектовано для хмарних бізнесів
Основний недолік	Складно використовувати, опції можуть здатися заплутаними	Неефективне управління інструментами	Обмежена кількість центрів даних по всьому світу

Кожен з цих провайдерів пропонує унікальні переваги та можливості, які можуть бути краще адаптовані до конкретних потреб вашого бізнесу. Важливо розглянути такі фактори, як вартість, географічне розташування дата-центрів, доступність сервісів, та специфічні вимоги до додатків, перш ніж вибирати хмарного провайдера.

Відповідно до опису [15] AWS (Amazon Web Services) є відмінним вибором для розгортання Zabbix з кількох причин:

1. Гнучкість та Масштабованість: AWS надає можливість легко масштабувати ресурси вгору або вниз залежно від потреб моніторингу, що є ідеальним для системи, як Zabbix, яка може вимагати зміни ресурсів у відповідь на зростаючий обсяг даних.

2. Широкий Вибір Сервісів: AWS пропонує широкий спектр сервісів, які можуть бути інтегровані з Zabbix для розширеного моніторингу, включаючи Amazon EC2, Amazon RDS, Amazon S3 та інші[16].

3. Безпека: AWS надає розширені можливості безпеки, які можуть бути використані для захисту інфраструктури Zabbix, включаючи шифрування даних, управління доступом та відстеження подій[17].

4. Надійність: Завдяки глобальній інфраструктурі та високому рівню доступності, AWS забезпечує стабільну платформу для неперервного моніторингу.

5. Інтеграція: AWS пропонує інтеграцію з Zabbix через HTTP, що дозволяє легко розгортати моніторинг без необхідності додаткових скриптів[16].

Використання AWS для Zabbix може значно спростити процес моніторингу, забезпечуючи ефективне відстеження ресурсів та додатків у хмарі.

1.4 Постановка задачі

Головним завданням є налаштування системи моніторингу мережевої інфраструктури для серверу і бази даних в хмарі. Зробивши аналіз вже існуючих систем моніторингу, було зроблено висновки щодо інструментів та метрик. В процесі функціонування системи необхідно діагностувати відхилення від штатного режиму функціонування системи та відповідно інформувати про проблему.

Для повноцінної реалізації проєкту потрібно виконати наступні задачі:

1. Проведення аналізу предметної області та визначення тенденцій та проблем в області моніторингу мережевої інфраструктури.
2. Проведення проектування інформаційної системи моніторингу.
3. Налаштування серверу та бази даних в хмарному середовищі.
4. Налаштування системи моніторингу та інформування про збої.
5. Проаналізувати отримані метрики функціонування системи.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ МОНІТОРИНГУ СЕРВЕРНОЇ ІНФРАСТРУКТУРИ

2.1 Вибір інструменту моніторингу

У сучасному світі корпоративних систем існує кілька популярних рішень щодо моніторингу. Серед них варто відзначити такі, як Prometheus, Datalog, Zabbix, Nagios та інші. Ці системи дозволяють відстежувати стан різних компонентів, ресурсів та послуг, що допомагає забезпечити безперебійну роботу бізнес-процесів.

Для бакалаврської роботи була обрано систему моніторінгу Zabbix з наступних причин:

- Всебічний Моніторинг: Zabbix є комплексним рішенням для моніторингу, яке може збирати метрики, виявляти проблеми, візуалізувати дані, сповіщати та надсилати повідомлення.
- Підтримка Різних Баз Даних: Zabbix може використовувати широкий спектр баз даних для зберігання даних, що робить його гнучким у виборі технологій.
- Масштабованість: Хоча Zabbix розроблений для невеликих та середніх середовищ, він також може бути оптимізований для великих установок[18].
- Вбудовані Функції Сповіщення: Zabbix має вбудовані можливості сповіщення, тоді як Grafana інтегрується з зовнішніми інструментами для сповіщень[15].
- Відкритий Вихідний Код: Zabbix є відкритим програмним забезпеченням, що дозволяє користувачам модифікувати та адаптувати його під свої потреби[18].
- Технічна Підтримка: Zabbix пропонує платні опції для підприємств, які можуть включати технічну підтримку та додаткові функції.

Zabbix є потужним інструментом для моніторингу, але, як і будь-яке програмне забезпечення, воно має недоліки, які можуть вплинути на його

використання в рамках поставленої задачі. Серед цих недоліків можна вважати значущими:

- Складність налаштування: Zabbix може бути складним для налаштування, особливо для нових користувачів або тих, хто не має досвіду з системами моніторингу
- Інтерфейс користувача: Деякі користувачі вважають інтерфейс користувача Zabbix застарілим або неінтуїтивним порівняно з іншими сучасними інструментами[19].
- Високі вимоги до ресурсів: Для великих установок Zabbix може мати високі вимоги до системних ресурсів, що може призвести до додаткових витрат на обладнання.
- Обмеження візуалізації: Хоча Zabbix пропонує візуалізацію даних, вона може бути не такою гнучкою або розширеною, як у деяких інших інструментів, таких як Grafana
- Сповіщення та повідомлення: Система сповіщень може бути складною для налаштування, і деякі користувачі можуть знайти її менш гнучкою порівняно з іншими рішеннями

Ці недоліки не обов’язково впливають на всіх користувачів однаково і можуть бути менш значущими залежно від конкретного використання та вимог до системи моніторингу.

2.2 Недоліки хмарної архітектури

Розглянемо в рамках вирішення поставленої задачі розгортання системи моніторингу Zabbix на Amazon Web Services (AWS), оскільки це хмарне рішення пропонує широкий спектр послуг, включаючи обчислення, зберігання, бази даних, мережі та аналітику.

Це дозволяє використовувати AWS для розгортання не лише Zabbix, а й інших додатків та сервісів(рис. 2.1).

Також AWS дозволяє легко масштабувати ресурси вгору або вниз в залежності від потреб системи моніторингу. Система забезпечує контроль потужності обчислювальних ресурсів, щоб відповідати навантаженню.

AWS має вбудовану інтеграцію з іншими сервісами, такими як Amazon CloudWatch, який може використовуватися для збору метрик та моніторингу ресурсів AWS.

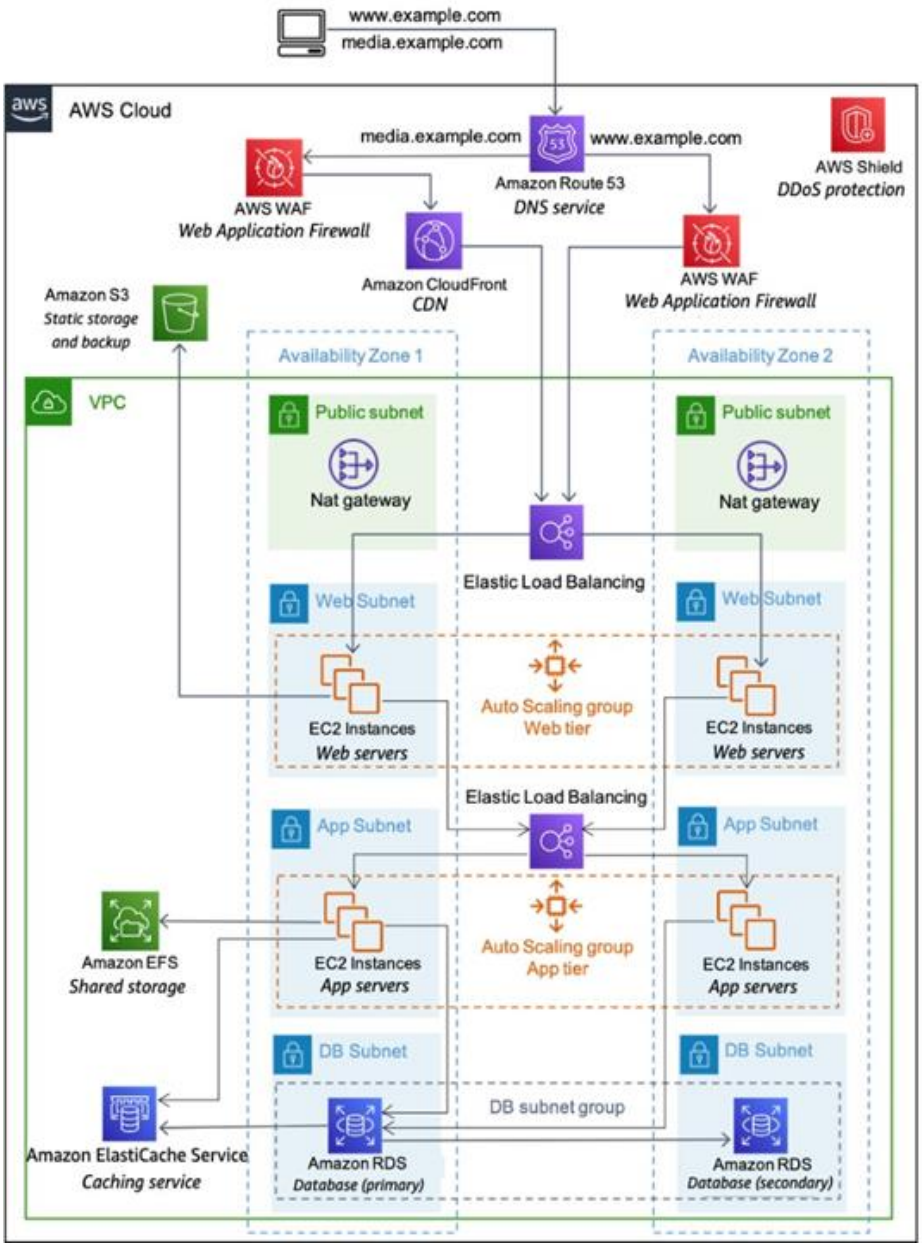


Рисунок 2.1 – Представлення класичної архітектури веб-додатків у хмарі AWS

З наведеного випливає що, використання AWS для розгортання Zabbix є раціональним вибором, оскільки це дозволяє отримати доступ до потужних ресурсів, високої доступності та гнучкості для системи моніторингу.

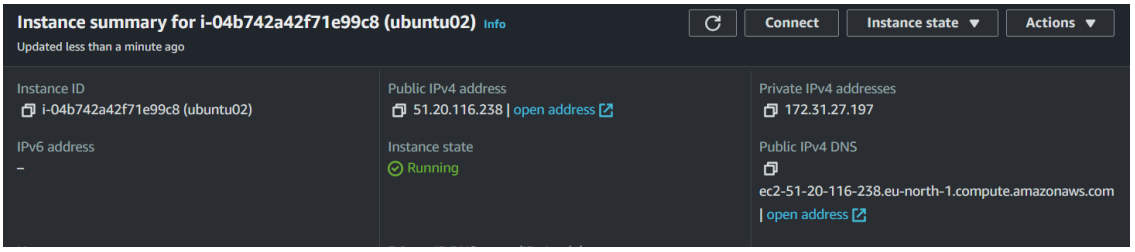
Але варто врахувати, що, як і будь-який інший продукт, AWS має свої недоліки:

- AWS обмежує ресурси за регіонами, що може стати проблемою, якщо потрібно більше ресурсів, ніж доступно в нашому регіоні[18].
- Спільне використання ресурсів: Якщо наша програма є високочутливою, той факт, що AWS є спільним ресурсом, може стати проблемою і у разі відмови серверного устаткування, яким опікується постачальник, робота нашої системи може бути порушена разом з іншими користувачами AWS[15].
- Безпека та конфіденційність: Хоча AWS надає високий рівень безпеки, критики висловлюють стурбованість щодо захисту резервних копій, витоку даних та проблем з конфіденційністю[16].

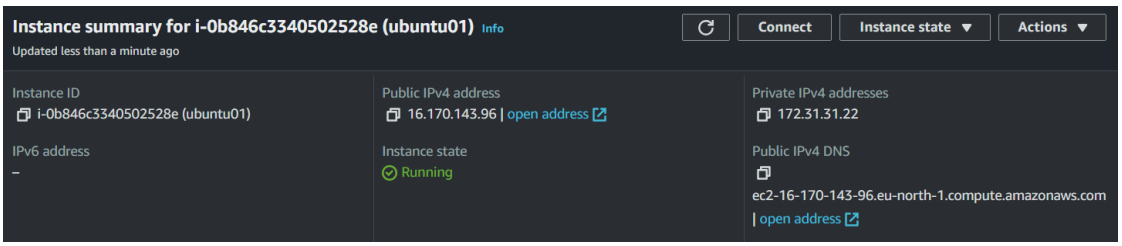
Ці недоліки варто враховувати при плануванні клієнт-серверної системи. І хоча перелічені вади можуть бути значними, вони не роблять AWS поганим вибором. Ці фактори враховані при виборі хмарного провайдера, та при плануванні архітектури системи, що розглядається.

2.3 Технічні характеристики програмного рішення

Реалізація системи передбачає дві окремо функціонуючі хмари AWS (Рис.2.2), які мають призначення: перша реалізує корпоративну систему з веб-ресурсів взаємодії з користувачем та відповідної корпоративної БД, друга – підтримує функціонуючу сутність Zabbix та його базу даних.



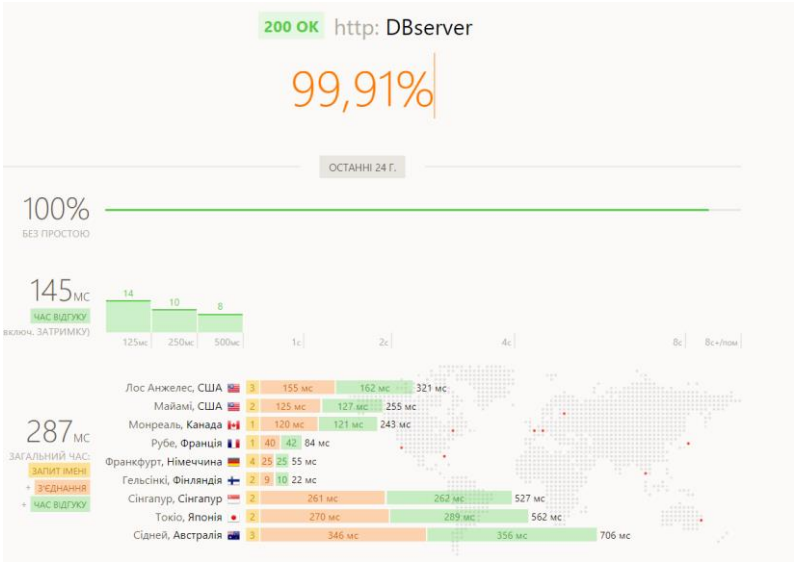
а



б

Рисунок 2.2 – Знімок характеристик хмарного середовища: а – AWS на платформі Ubuntu для корпоративного додатку; б – для середовища Zabbix.

У якості додаткового інструменту перевірки стану хмарних серверів розглянуто сервіс UpDown, який є безкоштовним опенсорсним додатком для перевірки доступності сервісів. Задача сервісу циклічно, відповідно за вказаним кроком перевірки, слідкувати за кодами відповіді серверів AWS (Рис.2.3) та у разі відмови сповіщати про проблему адміністратора. Сервіс вводиться як зовнішній інструмент оцінки роботи серверу Zabbix.



а

б

Рисунок 2.3– Знімок статистики відповідей AWS серверів:
а – сервер додатку; б – сервер Zabbix.

В загальному вигляді спроектована система має вигляд, представлений на концептуальній схемі (Рис.2.4) мережевої архітектури, де визначені основні елементи інфраструктури, як бази даних та сервіси, і суб'єкти взаємодії з середовищем, як клієнти та адміністратор.

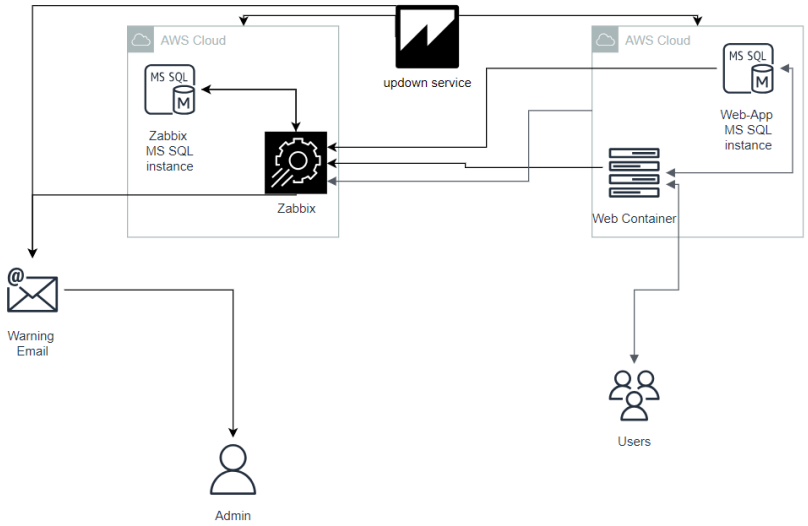


Рисунок 2.4 – Схема мережевої архітектури проекту.

Варто зазначити, що сутність адміністратора вказана, як суб'єкт, що отримує сповіщення про проблему. Це вимагає додаткового пояснення, оскільки в наведеній схемі виключено спектр дій адміністратора при виконанні процесів відновлення та переналаштування системи у разі відмови, оскільки ці процеси є складними для відтворення та можуть бути не стандартизовані, зумовлені різними причинами від дій користувача до відмов програмного чи апаратного

устаткування. Взаємодія ж клієнта з системою обмежена до роботи з корпоративним додатком, що відображено роботою з Web-контейнером.

2.4 Вимоги до проекту

Поставлена задача моніторингу серверної інфраструктури полягає в реалізації системи, яка якісно відтворює робочий режим корпоративного серверного рішення. Реалізація серверної архітектури не можлива без операційної системи. Приблизно 96.3% з топ-мільйона серверів у світі працюють на Linux[20]. Це система з відкритим вихідним кодом, яка характеризується своєю ефективністю та здатністю до швидкої роботи навіть на обмежених ресурсах. Ця операційна система є відкритою для користувачів, що дозволяє їм вносити зміни та адаптувати її до своїх потреб без необхідності вкладення значних ресурсів.

Відповідно до вимог сучасних операційних систем Unix/Linux, існує зростаюча потреба у використанні Bash, який є командним інтерпретатором, відомим також як оболонка. Bash надає можливість користувачам здійснювати командний ввід через термінал, автоматизувати рутинні завдання за допомогою скриптів, а також управляти файловою системою та процесами. Цей інструмент є незамінним для системного адміністрування та програмування, оскільки він дозволяє досягти глибокого рівня управління компонентами операційної системи. Для Bash існує програмна надбудова, котра називається fish. Вона значно покращує швидкість та якість роботи. Тому будемо використовавати саме fish.

Крім керування операційною системою необхідно реалізувати механізми взаємодії з користувачем в рамках скріптів мови Python. Це високорівнева, інтерпретована мова програмування з динамічною типізацією. Вона відома своєю читабельністю коду, простотою синтаксису та гнучкістю. Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтовану, функціональну та процедурну. Ті ж самі функції здатні виконвати shell скріпти.

Але вони не мають такої потужної функціональності, як мова програмування Python. Також їх важко підтримувати та читати.

Підтримка роботи веб-серверу з керуванням користувачами не можлива без відповідної бази даних, яка забезпечить зберігання та швидке опрацювання запитів. Існує два оптимальних рішення. MySQL та PostgreSQL. MySQL - це популярна система управління базами даних (СУБД), що працює на основі SQL (Structured Query Language). Вона є відкритим програмним забезпеченням та широко використовується для створення та управління реляційними базами даних. MySQL є надійним, масштабованим та ефективним рішенням для веб-додатків, онлайн-транзакцій та додатків, які потребують швидкого доступу до великих обсягів даних.

PostgreSQL — це високопродуктивна об’єктно-реляційна система керування базами даних (СКБД) з відкритим вихідним кодом. Вона підтримує розширений набір функцій SQL, транзакції з ACID-властивостями, які забезпечують надійність та цілісність даних, а також розширені можливості, такі як тригери, зовнішні ключі, матеріалізовані подання, і багато іншого. Основна характеристика чому PostgreSQL розглядається як альтернатива комерційним СКБД, як Oracle Database, Microsoft SQL Server, IBM DB2 – вільна система розповсюдження прорамного забезпечення.

За наведеним описом доступних СКБД переваги MySQL виводять її на позицію основного інструменту в роботі.

За описаною структурою виникає необхідність чітко визначити функціональні та нефункціональні вимоги до проекту, які визначають правила протікання процесів в системі та важливі аспекти реалізації цих процесів.

Функціональні вимоги:

1. Користувач може через додаток-клієнт(браузер) доєднатись до БД.
2. Мережевий трафік перевіряється системою моніторингу з невідкладним реагуванням на викиди

3. Система моніторингу виконує перевірку фізичної доступності устаткування; перевірка стану запущених служб; детальну перевірку

продуктивності, завантаження; та реагує на відхилення в роботі відправкою повідомлень через sms та email.

4. Система виконує регулярне створення бекапів.

5. Скрипт для зберігання даних повинен мати можливість підключатися до бази даних та вставляти дані відповідно до заданої схеми.

6. Скрипт для перевірки та додавання в файл повинен записати дані і потім провалідувати дані: Вік, Стать, Зарплата.

7. Скрипт, який запускається, повинен перевірити файл на наявність специфічного повідомлення. На виході маємо, нуль або одинцю. Якщо вилітає нуль або виключення, далі спрацьовує тріггер для відповідної реакції моніторінг системи.

8. Передбачено процесс Poller, який виконує збір інформації щодо ОС та мережевого обладнання.

9. Zabbix створює при першому запуску, та регулярно оновлює “мапу мережі”.

Нефункціональні вимоги:

1. Всі скрипти повинні мати рівень надійності та забезпечувати точність обробки даних, визначені метриками: час безвідмовної роботи: 23 години, середній час до відмови: не більше 1 хвилини, ймовірність відмови протягом 10 годин: 1%

2. Система повинна бути здатна працювати для Linux Ubuntu 18.04 та вище та версіях Python 3+

3. Регулярне оновлення мапи мережі відбувається кожну добу.

4. Максимальна кількість одночасних користувачів мережею 20.

5. Максимальна кількість запитів до БД 100/хвилину.

6. Час відгуку скриптів не повинен перевищувати 2 секунди.

7. Розсилка email виконується через сервіс Google SMTP.

8. Моніторинг використання і навантаження процесора, завантаження мережі, завантаженні і використанні каналу зв'язку реалізовано через Zabbix Agent.

9. Перевірка на доступність і реакцію сервісів SMTP та HTTP реалізовано через Simple Checks.

10. Бекапи створюються кожні 7 днів.

11. Параметри сервера: 1 vCPU Core Icon, 4 GB RAM, 50 GB NVMe дискового простору.

12. Прийнятною для роботи системи є температура в 40-60 градусів.

10. При заповненні даних Вік: є додатним числом і не перевищує максимального значення 150 років.

11. При заповненні даних Стать відповідає одному з варіантів: ‘Male’, ‘Female’ або ‘Other’.

12. При заповненні даних Зарплата є додатним числом і не перебільшує 1 мільйона.

2.5 Забезпечення якості проектного рішення

В рамках створення дизайну серверної системи та плану моніторингу процесів варто визнчити ряд тест-кейсів, які дозволять регулярно перевіряти відповідність реалізації проекту, визначеним в рамках планування, вимогам, як функціональним так і нефункціональним. Крім того наявність тестів гарантує легший процес пошуку потенційних вад при внесенні змін в хід роботи програних рішень.

Тест-кейси для скрипта зберігання даних, скрипта перевірки та додавання в файл, для повторюваного для скрипта, що запускається кожні 5 хвилин, перевірки часу відгуку та перевірки надійності подані в таблиці 2.1.

Таблиця 2.1– Тест-кейси перевірки скриптів.

№	Мета	Кроки відтворення	Очікуваний результат
1	Перевірити здатність скрипта зберігати дані в базу даних.	1. Запустити скрипт із валідними даними для вставки. 2. Зробити запит до бази даних, щоб перевірити чи додались дані	Дані успішно збережені в базі даних.

№	Мета	Кроки відтворення	Очікуваний результат
		SELECT * FROM my_table LIMIT 1;	
2	Перевірити здатність скрипта валідувати та записувати дані	1.Запустити скрипт із невалідними даними. 2. Перевірити, чи дані не були додані до файлу. - відкрити вміст my_data.csv	Невалідні дані не були додані до файлу
3	Перевірити здатність скрипта ідентифікувати позитивний результат.	1.Запустити скрипт із файлом, що містить негативний результат. 2. Чекати 5 хвилин. Перевірити чи була виконана відповідна дія подивившись в my_data.csv файл	Скрипт ідентифікував негативний результат агент системи моніторінгу считав дані та стригерів alert
4	Перевірити, чи час відгуку скриптів не перевищує 2 секунди	1. Запустити кожен скрипт із валідними даними. Виміряти час відгуку.	Час відгуку кожного скрипта не перевищує 2 секунди.
5	Перевірити надійність скриптів при тривалій роботі.	1.Запустити кожен скрипт на 24 години. 2.Перевірити, чи не було помилок або збоїв подивишись логі системи моніторінгу.	Скрипти працювали стабільно та без помилок протягом усього періоду. Якщо помилки були, вони будуть там відображені.

В коді проекту заплановано три функції: data_storage, data_check_add та periodic_check, які відповідають задачам зберіганням даних в БД, валідації даних та періодичної перевірки. В роботі використовується модуль unittest для створення тестів для кожного з скриптів.

3 НАЛАШТУВАННЯ МОНІТОРИНГУ

3.1 Розгортання клієнт-серверної системи

Хмарна платформа AWS від Amazon пропонує декілька рішень для роботи з серверами. Для цього проекту підходять t2.micro та вище.

Відомо що платформа AWS пропонує, free trial, іншими словами безкоштовну версію деяких серверів для роботи з серверами. t2.micro - це варіант серверу Amazon EC2 котрий вважається підходячим для загального використання. Цей екземляр має 1 віртуальний процесор (vCPU), 1 гігабайт оперативної пам’яті та доволі помірну продуктивність для використання в мережі . Він нормально показує себе при використанні невеликих застосунків, по типу мікросервісів, бази даних невеликого об’єму, відтворення та тестування середовищ з різними операційними системами, локальні репозиторії для програмного коду, тестування прототипів продукту.

Але його ресурсів не вистачить для повноцінної роботи Zabbix Server, тому ми його використаємо для тестового серверу. Наступним дослідним варіантом став EC2 t3a.medium. Сутність t3a.medium - це варіант серверу Amazon EC2 котрий також вважається популярним для використання в системах загального призначення. Цей екземпляр має 2 віртуальних процесори) та 4,0 гігабайти RAM. T3a.medium має призначення для керованого використання ресурсів процесору, але може при потребі збільшити виробничу продуктивність. Він демонструє найкращі показники продуктивності при використанні невеликих застосунків, як мікросервіси, бази даних невеликого об’єму, відтворення та тестування середовищ з різними операційними системами, локальні репозиторії для програмного коду, тестування прототипів продукту та інших. Перевагою такої конфігурації є можливості буста, та більший об’єм оперативної пам’яті.

Докладніше розглянемо алгоритм запуску та підключення до серверу.

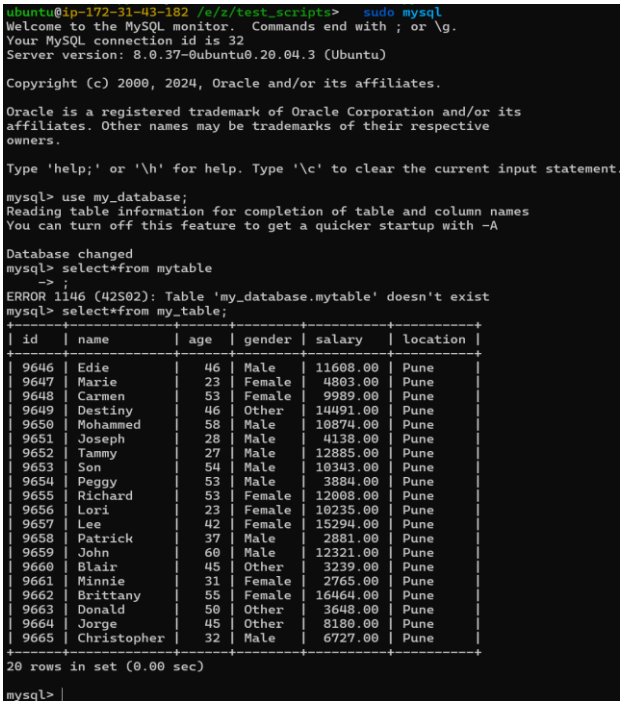
Процедура запуску Instance EC2 починається з входу до AWS Management Console, за допомогою якої створюється новий EC2 за обраною операційною системою. В роботі розглянуто використання Ubuntu 20.04 LTS, яка є стабільно

працюючою системою з актуальними, стабільними версіями програмного забезпечення, яке не породжує конфлікти версій.

Стандартне підключення до серверу, "із коробки", іде через key/value pair. Втім, для зручності ці параметри варто змінити з додаванням можливості підключатись за паролем.

У якості необхідного ПЗ, для виконання поставленої задачі, в систему встановлюється інтерактивна командна оболонка fish, яка є надбудовою стандартною оболонкою bash в операційній системі Linux, котра суттєво полегшує працю з командним рядком.

Під час розробки програмних додатків реального світу є поширена практика дотримуватись актуальних, але не найновіших, версій операційних систем та додатків. Галузевим стандартом є правило розгортати те, що вже показало себе як стабільно працююча система, і не породжує конфліктів. Виходячи з цієї позиції, в роботі розглянуто використання версії операційної системи Linux Ubuntu 20.04, що стабільно працює в поєднанні з Zabbix 6.0 з Apache та БД MySQL (рис.3.1).



```
ubuntu@ip-172-31-43-182 /e/z/test_scripts>   sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.37-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use my_database;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select*from mytable
    -> ;
ERROR 1146 (42S02): Table 'my_database.mytable' doesn't exist
mysql> select*from my_table;
+------+-------------+------+--------+----------+----------+
| id   | name        | age  | gender | salary   | location |
+------+-------------+------+--------+----------+----------+
| 9646 | Edie        |   46 | Male   | 11608.00 | Pune     |
| 9647 | Marie       |   23 | Female |  4803.00 | Pune     |
| 9648 | Carmen      |   53 | Female |  9989.00 | Pune     |
| 9649 | Destiny     |   46 | Other  | 14491.00 | Pune     |
| 9650 | Mohammed    |   58 | Male   | 10874.00 | Pune     |
| 9651 | Joseph      |   28 | Male   |  4138.00 | Pune     |
| 9652 | Tammy       |   27 | Male   | 12885.00 | Pune     |
| 9653 | Son         |   54 | Male   | 10343.00 | Pune     |
| 9654 | Peggy       |   53 | Male   |  3884.00 | Pune     |
| 9655 | Richard     |   53 | Female | 12008.00 | Pune     |
| 9656 | Lori        |   23 | Female | 10235.00 | Pune     |
| 9657 | Lee         |   42 | Female | 15294.00 | Pune     |
| 9658 | Patrick     |   37 | Male   |  2881.00 | Pune     |
| 9659 | John        |   60 | Male   | 12321.00 | Pune     |
| 9660 | Blair       |   45 | Other  |  3239.00 | Pune     |
| 9661 | Minnie      |   31 | Female |  2765.00 | Pune     |
| 9662 | Brittany    |   55 | Female | 16464.00 | Pune     |
| 9663 | Donald      |   50 | Other  |  3648.00 | Pune     |
| 9664 | Jorge       |   45 | Other  |  8180.00 | Pune     |
| 9665 | Christopher |   32 | Male   |  6727.00 | Pune     |
+------+-------------+------+--------+----------+----------+
20 rows in set (0.00 sec)

mysql>
```

Рисунок 3.1 – Таблиця досліджуваного тестового додатку

Після налаштування Zabbix Server необхідно налаштувати Test Server, з використанням вимог до устаткування як для серверу моніторігу за деякими виключеннями, а саме – для тестового серверу немає потреби встановлювати всі компоненти Zabbix, достатньо лише Zabbix Agent.

Після встановлення агента необхідно відредагувати zabbix_agentd.conf файл (рис.3.2) так, щоб в параметрі Server використати IP адресу Zabbix Server, якій налаштовано в системі. Параметр ListenPort змінити з 10050 на 10051, оскільки за цим портом Zabbix agent слухає Zabbix trapper. Параметр ServerActive має бути таким самим як і Server, а Hostname має співпадати з тим що вказано для Zabbix Frontend.

```
##### Active checks related

### Option: ServerActive
#       List of comma delimited IP:port (or DNS name:port) pairs of Zabbix servers and Zabbix
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host is specified.
#       If port is not specified, square brackets for IPv6 addresses are optional.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=127.0.0.1,34.229.135.49

### Option: Hostname
#       Unique, case sensitive hostname.
#       Required for active checks and must match hostname as configured on the server.
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
Hostname=Tested server


### Option: HostnameItem
#       Item used for generating Hostname if it is undefined. Ignored if Hostname is defined.
#       Does not support UserParameters or aliases.
```

Рисунок 3.2 – Налаштування zabbix_agentd.conf

На Рисунку 3.3 демонструється вміст директорії test_scripts з набором кодів Python необхідних для роботи з Базою Даних додатку та Zabbix.

```
ubuntu@ip-172-31-43-182 /e/z/test_scripts> pwd
/etc/zabbix/test_scripts
ubuntu@ip-172-31-43-182 /e/z/test_scripts> ls
insert_script.py*  my_data.csv  write_to_csv_file.py*
ubuntu@ip-172-31-43-182 /e/z/test_scripts>
```

Рисунок 3.3 – Зображення вмісту test_scripts

Останнім кроком налаштуємо Zabbix Frontend. Для цього необхідно перейти на сторінку адміністрування за URL з IP адресою серверу AWS та ендпоінтом zabbix. В розглянутій системі цей шлях описує адреса "http://34.229.135.49/zabbix/".

Після авторизації в акаунт адміна, в секції мені Configuration небхідно створити хост, "Create host" – названий відповідно до конфігурації описаної вище, "Tested server" для групи Zabbix servers. Рисунок 3.4 демонструє додавання в Interfaces сутності Agent, який використовує публічну IP адресу та порт 10051 для zabbix trapper.

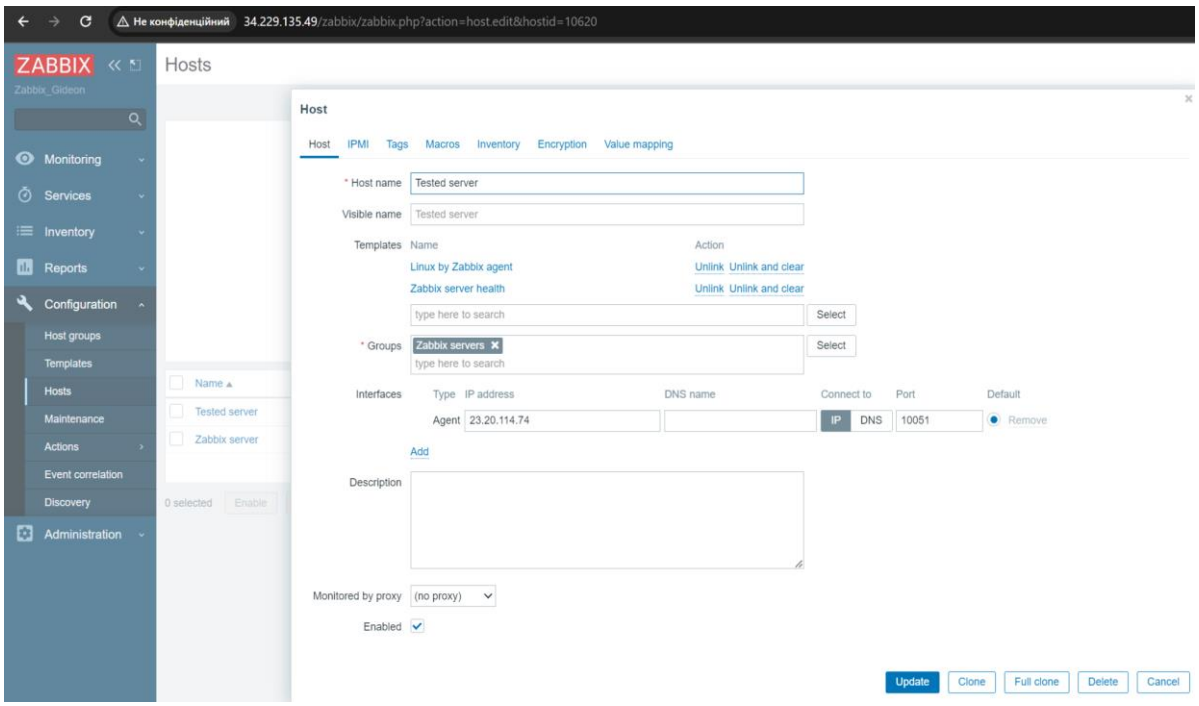


Рисунок 3.4 – Сторінка налаштування Zabbix Frontend

3.2 Моніторинг доступності

В умовах клієнт-серверної взаємодії в корпоративних системах критично важливо щоб сервер працював безвідмовно. Відповідно якщо трапляється позаштатна ситуація в роботі системи – необхідно інформувати про проблему в найкоротші терміни. Для цього в розглянутій системі використовується система Zabbix Тригерів та нотіфікацій. У якості механізму інформування часто

використовують технології розсилки сповіщень через SMS, email, Discord, Slack, PagerDuty, тощо.

В дослідженні відслідковувти доступність серверу вирішено за допомогою Zabbix server health. Найпоширінеша практика це Zabbix agent ping (рис.3.5). Він повертає 1 якщо все добре, та нічого не повертає, якщо агент не доступний. Якщо трапляється останнє, зазвичай це значить що сервер не доступний, відповідно потрібно втручання адміністратора.

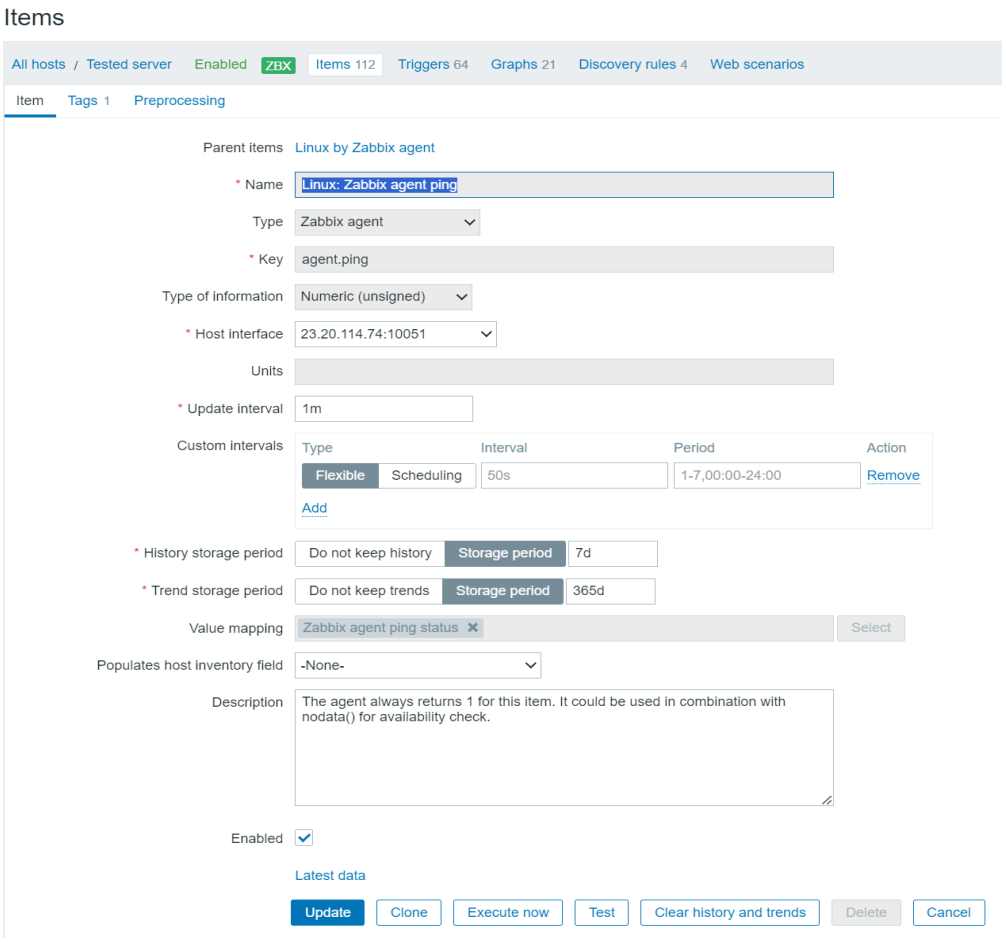


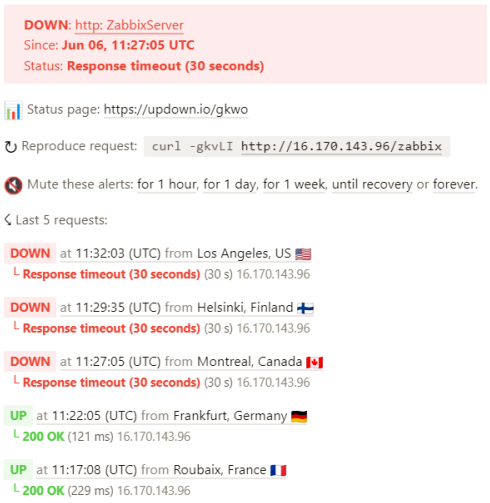
Рисунок 3.5 – Налаштування Zabbix agent ping

Оскільки моніторинг доступності є значним елементом управління мережевою IT-інфраструктурою, оскільки дозволяє оперативно виявляти та вирішувати проблеми, які можуть вплинути на безперебійну роботу сервісів. Очевидні переваги моніторингу, як виявлення проблем у реальному часі, підвищення надійності та стабільності, підтримка високого рівня

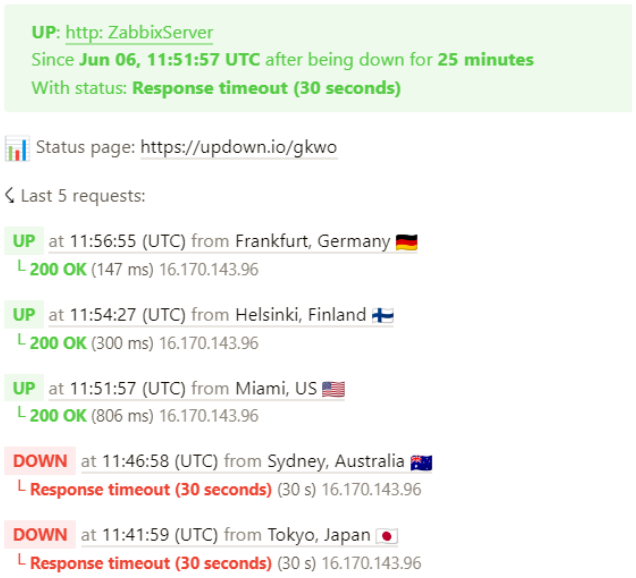
обслуговування, чи спрощена аналітика та звітність – є вагомими, втім реалізують скоріше самозамкнуту систему. Два хмарні середовища від одного постачальника, одне з корпоративним додатком – інше з середовищем діагностики все ще сильно залежні від устаткування, мережі чи самого постачальника.

Доцільним рішенням є додатковий моніторинг за допомогою іншого інструмента, як сервіс UpDown, який забезпечить резервний моніторинг з інакшим підходом та незалежними перевірками до основного інструменту. Таке рішення є потенційним зниженням ризиків, оскільки представлятиме зовнішній інструмент перевірки обох хмар.

Під час налаштування обох хмарних складових розглянутої в роботі системи очевидно були процеси перезавантаження серверів чи їх переналаштування, яке заважало коректно відповісти на запити UpDown. Така поведінка середовища зручно була відображена в вчасних звітах відмов, що описували критичні для користувача доступи (для серверу корпоративного додатку) та потенційні проблеми розподілу ресурсів (для серверу моніторингу під час його налаштування). Це, в свою чергу, дозволило вчасно вирішити питання кешування даних Zabbix. На рисунку 3.6 проілюстровано приклади звітів відмови та відновлення роботи серверу, який розсилається у випадку проблеми адміністратору системи.



а



б

Рисунок 3.6 – Звіти листів сервісу UpDown щодо роботи серверу.
а – інформування про відмову, б – сповіщення про відновлення роботи.

Таким чином в системі було досягнуто перевірку сертифікатів SSL та аналіз часу відгуку веб-сторінок з різних географічних локацій. В той час, як Zabbix може моніторити внутрішні параметри сервера, резервна перевірка Updown здійснює зовнішні перевірки доступності через HTTP/HTTPS, що в свою чергу дозволяє отримати більш повну картину стану системи.

3.3 Моніторинг трафіку

Zabbix дозволяє моніторити значний набір аспектів роботи сервера, одним з них є мережевий трафік. Zabbix дозволяє аналізувати зовнішній трафік у реальному часі та допомогає виявити патерни поведінки, що дозволяє виявляти атаки на ранніх стадіях.

Для корпоративної серверної системи доречно відслідковувати кількість пакетів котрі передаються на сервер з помилками. Налаштований в системі елемент net.if.in та тригер інформування про проблему (рис.3.7) прцюють відповідно до параметрів zabbix_agentd.conf, досліджуючи вхідний трафік на інтерфейсі.

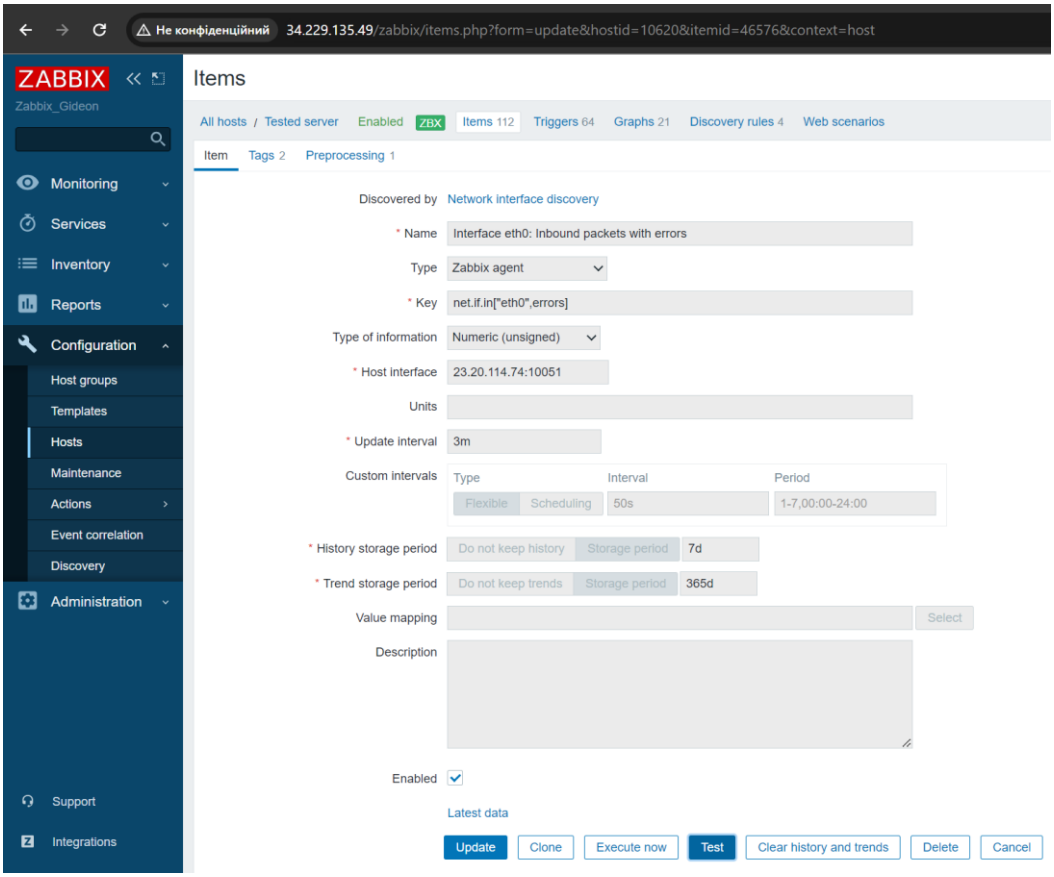


Рисунок 3.7 – Налаштування елементу даних, відповідального за збір вхідного трафіку.

Досліджуючи пакети, які надходять в мережу, Zabbix також здатен аналізувати log-files та log-events,що дозволяє виявляти аномалії та потенційні загрози ще на початку виникнення проблем. У разі виявлення підозрілої активності чи поведінки, що відповідає шкідливим шаблонам – спрацює тригер, котрий повідомить Zabbix Адміністратора про проблему.

3.4 Моніторинг внутрішніх параметрів та ресурсів серверу

Наступним ключовим показником роботи системи є CPU Utilization, метрика, котра висвітлює відсоток часу, під час якого процесор зайнятий обробленням інструкцій отриманих від серверу. Наведена метрика є одним з найважливіших показників здоров’я сервера. Галузевим стандартом вважається відправляти повідомлення адміністратору, коли CPU Utilization достигає 90%. Якщо CPU завантажено на 100%, то відмова серверу неминуча, і зазвичай

допоможе тільки перезавантаження. Рисунок 3.8 демонструє налаштування Zabbix для моніторингу використання потужностей процесора.

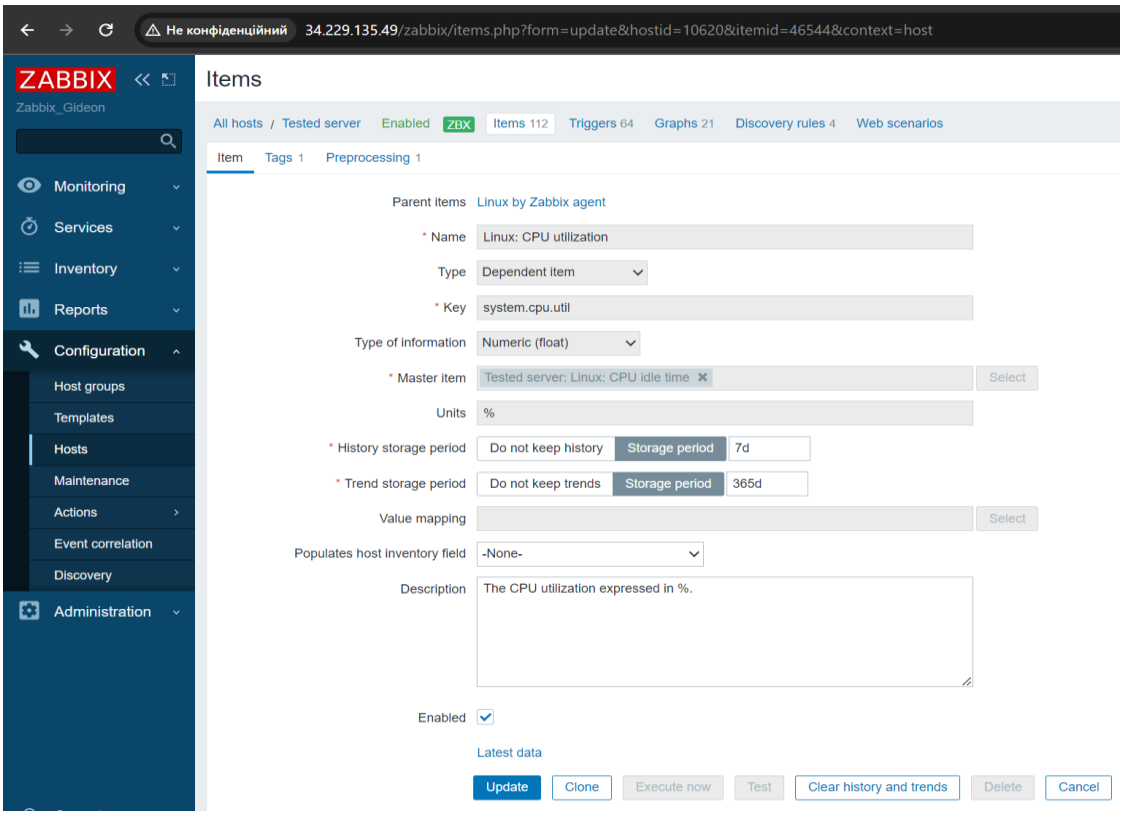


Рисунок 3.8 – Налаштування моніторингу ресурсів процесора

Серед поширених практик в галузі при використанні Linux є команда uptime, яка має відображати інформацію в командному рядку, яка буде містити поточний час, час безперервної роботи системи, кількість юзерів, які наразі використовують систему, та середнє навантаження.

В розгорнутій відповідно до проекту описаної серверної інфраструкрути системи Zabbix реалізовано виклик item system.uptime (рис.3.9), який циклічно виконує заміри з часом 30 секунд та повертає загальну кількість секунд безперевної роботи серверу. Якщо це значення замале, це значить що система постійно перезавантажується і потребує негайного втручання адміністратора

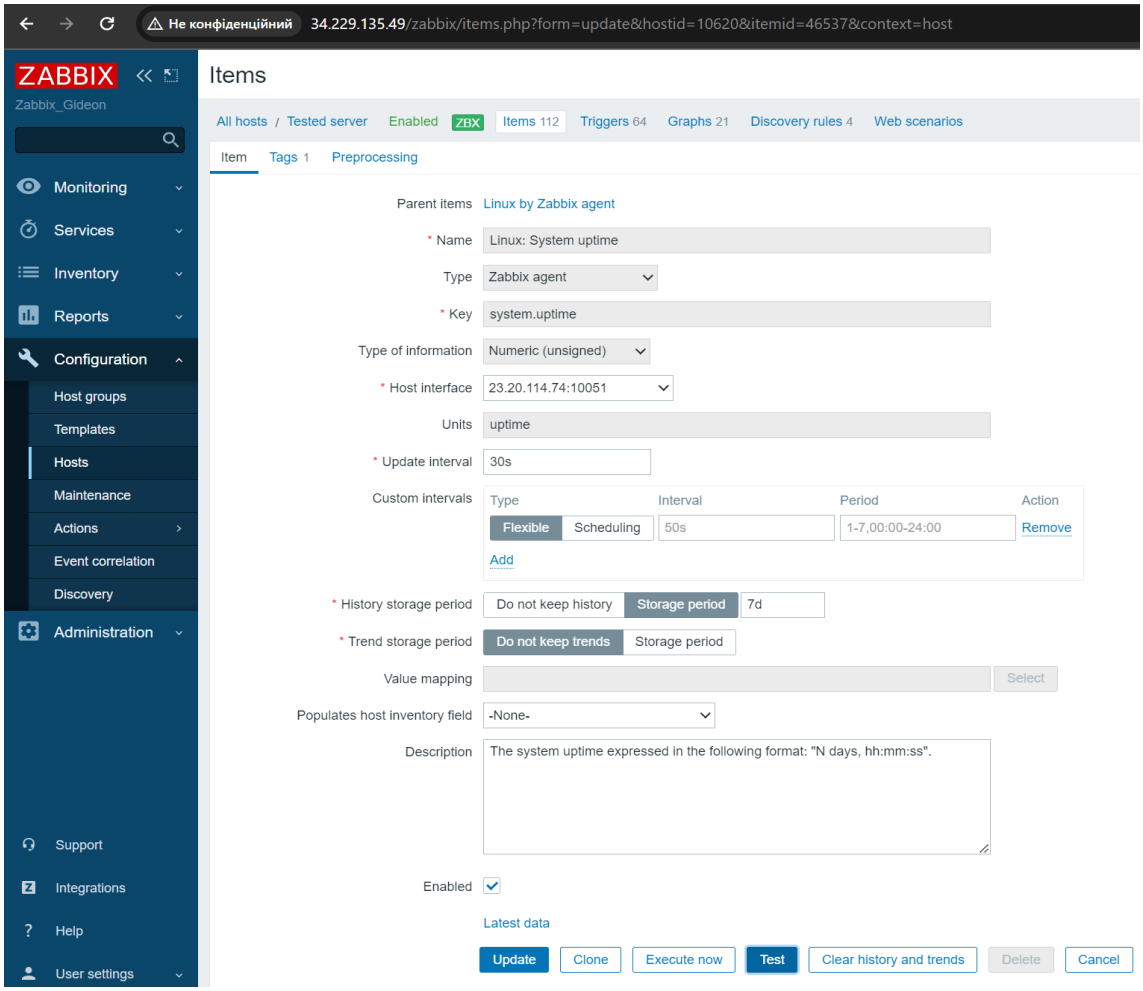
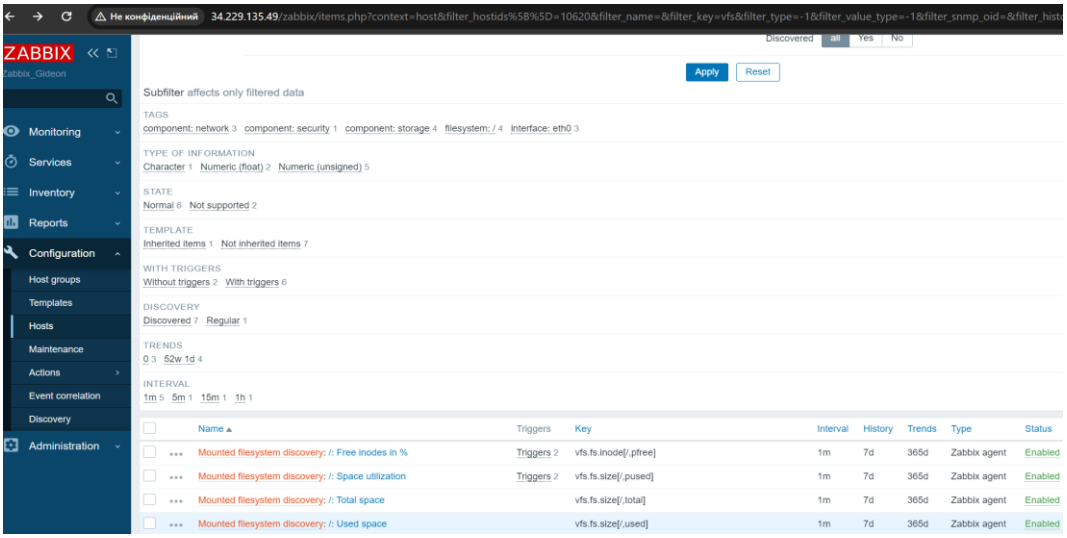


Рисунок 3.9 – Виклик перевірки часу безперервної роботи серверу

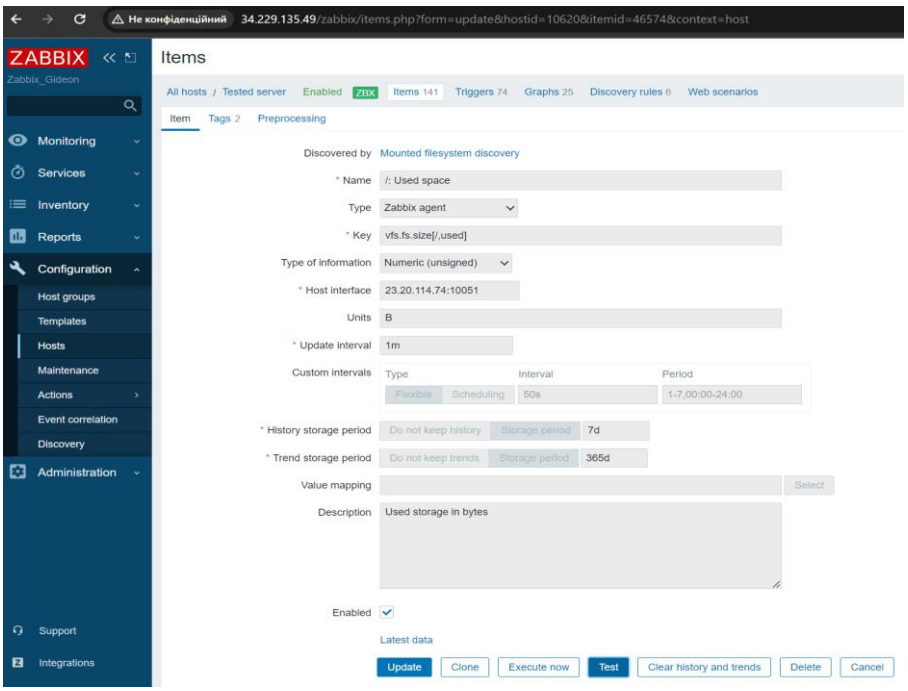
Наступний критичний за важливісттю показник – робота ROM та RAM пам’яті на стороні серверу. Проблеми з використання пам’яті можуть стати причиною відмови додатку, що веде до втрати ефективної роботи з клієнтами, і відповідно шкодить системі замовника.

Хмари AWS зручні тим, що система дозволяє швидко масштабуватись, якщо виробничих ресурсів серверу вже не вистачає. Втім рішення зміни ресурсів та тарифного плану необхідно обгрунтувати за доцільністтю та ефективним розподілом ресурсів системи. Звідси випливає необхідність регулярної перевірка, що забезпечить наступні метрики: якщо доступна пам'ять менше 20% від загальної, або використання пам'яті перевищує 80% – надсилається попередження для адміністратора. Розглянемо наступне налаштування моніторингу (рис.3.10) елементів даних системи:

- vm.memory.size[total] – загальна кількість фізичної пам'яті.
- vm.memory.size[available] – доступна пам'ять.
- vm.memory.size[used] – використана пам'ять.
- vfs.fs.size[/,total] – загальний розмір файлової системи (ROM).
- vfs.fs.size[/,used] – використана частина файлової системи (ROM).



а



б

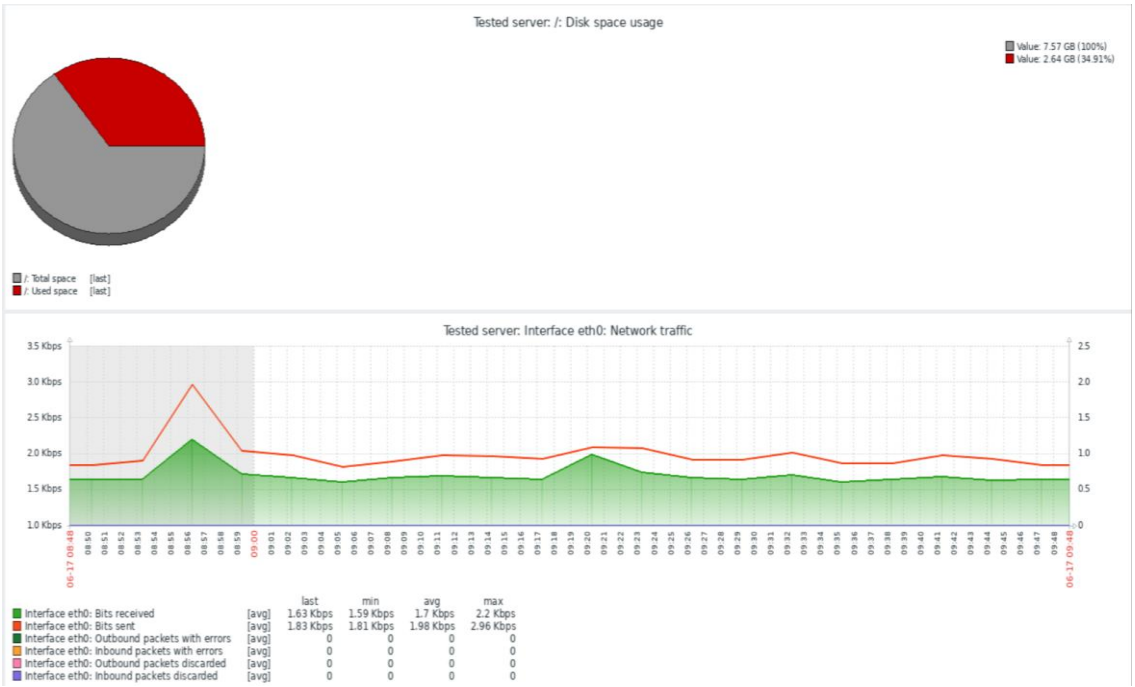
Рисунок 3.10 – Налаштування дослідження ресурсів пам'яті системи: а – дисковий простір; б – дослідження зайнятої пам'яті в байтах

Метріки total space та used space дозволяють відслідкувовувати скільки пам’яті використовується та скільки вільної пам’яті залишилось. Якщо додаток працює некоректно з пам'яттю, або потрібне масштабування для зміни потужностей системи – наведене налаштування тригерів допоможе відслідкувати та повідомити адміністратору коли потрібне втручання.

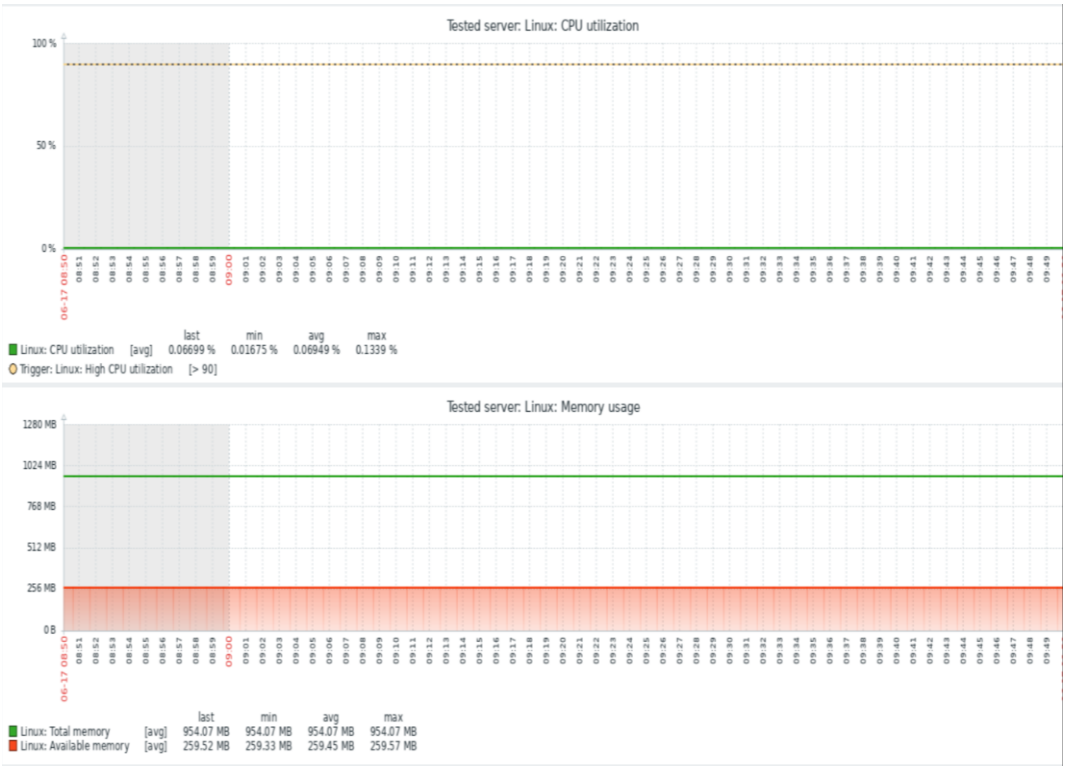
3.5 Аналіз результатів

Під час реалізації спроектованої системи були отримані результати моніторингу системи в робочому режимі.

За допомогою Zabbix було отримано графіки (рис.3.11) моніторингу дискового простору, навантаження процесору та пам'яті. Ці перевірки виконані в рамках моніторингу ресурсів серверу та моніторингу БД, з перевіркою коректності роботи скриптів.



а



б

в

Рисунок 3.11 – Графіки Zabbix моніторингу: а – використання дискового простору, б – ресурси процесора та пам'яті, в – керування процесами.

Аналізуючи графіки на Рисунку 3.11 можна бачити, що сервер в штатному режимі БД використовує приблизно 2,64 ГБ дискового простору, що є 34.91% від загального простору. Використання пам'яті в системі сягає від 72,8%, що

лишає лише 259,5 МБ на вільну пам'ять та буфери/кеш, які можуть бути використані для нових процесів. Втім, використання ресурсів процесору в середовищі серверу БД не значне, близько 0,07%.

Перевірка доступності сервера за допомогою Zabbix ping agent визначає наступні показники як ключові:

Відсоток часу доступності, показує, скільки часу сервер був доступний за визначений період – 96.3%, що є високим показником для системи.

Середній час відповіді, який потрібен серверу для відповіді на запит – 81 мс. Час відповіді зазвичай коливається, оскільки є чутливим до кількості запитів та користувачів в системі, втім отримані результати визначають доволі високу швидкість відповіді.

Кількість відмов, коли сервер був недоступний за досліджуваний проміжок сягнув значення 2. Загалом за досліджуваний проміжок отримано малу кількість відмов, які в свою чергу зумовлені збільшенням навантаження на сервер БД через розгортання там інформаційних сторінок, а саме додаткової інсталяції php середовища та контейнеру Apach, а також при стрес-тесті доступу до серверу.

Аналізуючи отримані результати, можна стверджувати, що спроектована система успішно розгорнута в середовищі. Всі сутності, визначені дизайном, виконують свої функціональні вимоги, дотримуючись визначених показників функціональності. В системі спостерігається значне використання пам'яті, що зумовлене архітектурним рішенням – використанням t2.micro сервером для БД, втім за рештою метрик, отриманих в рамках дослідження серверної інфраструктури, система показала значну швидкодію та ефективне керування внутрішніми процесами. Система інформування про відхилення від штатного режиму роботи продемонструвала швидке реагування на тригери та змістовне і вчасне повідомлення адміністратора про необхідність втручання.

ВИСНОВКИ

Кваліфікаційну роботу бакалавра можна вважати успішною, оскільки в рамках її виконання були вирішені всі поставлені задачі. В ході роботи було сформовано серверну архітектуру та налаштовано середовище моніторингу важливих вузлів мережевого середовища. Функціональне, ефективне, стабільне та завадостійке серверне середовище покладається на значну кількість нюансів в роботі, які неможливо передбачити в процесі реального використання серверних ресурсів користувачами, втім система діагностики стану серверної інфраструктури на основі мережевого моніторингу забезпечує інструмент адміністрування, який полегшує процес виявлення вад системи та пришвидшує відновлення у разі відмови.

В ході роботи було виконано:

1. Проведено аналіз предметної області та визначенно тенденції та проблеми в області моніторингу мережевої інфраструктури. Виконано порівняння існуючих рішень та програмних засобів діагностування збоїв та відхилень від штатного режиму роботи. За результатами аналізу визначено Zabbix, як основний інструмент виконання моніторингу.

2. Проведено проектування інформаційної системи моніторингу. Відповідно до галузевих стандартів визначено серверне середовище та виконано проектування серверної інфраструктури та мережевого устаткування, визначено важливі вузли, метрики які необхідно забезпечити та сформовано вимоги до функціонування системи.

3. Налаштовано сервер та MySQL базу даних в хмарному середовищі AWS t2.micro, t3a.medium на платформі Linux: Ubuntu. Інстальовано програмне забезпечення для забезпечення користувацького доступу до веб-ресурсів. Сформовано інформаційні веб-сторінки, для виводу адміністративної інформації. В середовище додані скріпти для роботи з MySQL БД та забезпечення користувацької взаємодії з середовищем.

4. Налаштовано систему моніторингу Zabbix та інформування про збої. Як додатковий інструмент моніторингу доступності розглянуто ресурс updown для ведення збору статистики функціонування серверу БД і Zabbix. Реалізовані скріпти для інформування адміністратора про відхилення від штатного режиму роботи системи та про явні відмови.

5. Проаналізовано отримані метрики функціонування реалізованої системи. Визначено та опрацьовано випадки неефективного використання ресурсів. Сформовано основні параметри метрик, які дозволять швидше оцінити ваду в роботі та полегшити адміністрування середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Real User Monitoring [Електронний ресурс]. – Режим доступу: https://cronitor.io/guides/real-user-monitoring (дата звернення 05.06.2024).

2. AI Network Monitoring [Електронний ресурс]. – Режим доступу: https://research.aimultiple.com/ai-network-monitoring/ (дата звернення 05.06.2024).

3. What is System Monitoring? [Електронний ресурс]. – Режим доступу: https://blog.krutus.com/what-is-system-monitoring/ (дата звернення 05.06.2024).

4. SIEM Use Cases [Електронний ресурс]. – Режим доступу: https://www.exabeam.com/explainers/siem-security/siem-use-cases/ (дата звернення 05.06.2024).

5. What is High Availability? [Електронний ресурс]. – Режим доступу: https://phoenixnap.com/blog/what-is-high-availability (дата звернення 05.06.2024).

6. The Ultimate Guide to Database High Availability [Електронний ресурс]. – Режим доступу: https://www.percona.com/blog/the-ultimate-guide-to-database-high-availability/ (дата звернення 05.06.2024).

7. N. Sukhija, E. Bautista et al. Event Management and Monitoring Framework for HPC Environments using ServiceNow and Prometheus. MEDES '20: Proceedings of the 12th International Conference on Management of Digital EcoSystems November 2020, pp.149–156. DOI: 10.1145/3415958.3433046

8. N.Sarwar, J.N. Qureshi, T. Iqbal et al. Analysis of Web Monitoring Servers and Tools for Cloud Computing Services. In: Jawawi, D.N.A., Bajwa, I.S., Kazmi, R. (eds) Engineering Software for Modern Challenges. ESMoC 2021. Communications in Computer and Information Science, vol 1615. Springer, Cham. DOI: 10.1007/978-3-031-19968-4_4

9. F. Mohd , M. Mohd et al. Performance Analysis of Open-Source Network Monitoring Software in Wireless Network. Journal of Computing Research and Innovation, 8(2), 31–44. DOI:10.24191/jcrinn.v8i2.375

10. S. Jena, S. Mohapatra, et al. Open-Source Cloud Infrastructure & Application Monitoring and Alerting System. NeuroQuantology; Bornova Izmir Vol. 20, Iss. 10, (2022). pp. 7909 - 7916. DOI:10.14704/nq.2022.20.10.NQ55779

11. Платформа AWS (Amazon Web Services) [Електронний ресурс]. – Режим доступу: https://wiseit.com.ua/services/hmarni-rishennya/platforma-aws-amazon-web-services/ (дата звернення 05.06.2024).

12. Gupta, Bulbul & Mittal, Pooja & Mufti, Tabish. (2021). A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services. doi:10.4108/eai.27-2-2020.2303255.

13. M. Saraswat and R. C. Tripathi, "Cloud Computing: Comparison and Analysis of Cloud Service Providers-AWs, Microsoft and Google," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 2020, pp. 281-285, doi: 10.1109/SMART50582.2020.9337100.

14. Kamal, Muhammad Ayoub & Raza, Hafiz & Alam, Muhammad & Mazliham, M.. (2020). Highlight the Features of AWS, GCP and Microsoft Azure that Have an Impact when Choosing a Cloud Service Provider. International Journal of Recent Technology and Engineering (IJRTE). 8. doi: 10.35940/ijrte.D8573.018520.

15. Zabbix Documentation [Електронний ресурс]. – Режим доступу: https://www.zabbix.com/documentation/1.8/en/manual (дата звернення 06.06.2024).

16. AWS Integrations [Електронний ресурс]. – Режим доступу: https://www.zabbix.com/integrations/aws (дата звернення 05.06.2024).

17. Should I Still Use Zabbix in AWS? [Електронний ресурс]. – Режим доступу: https://www.sios-apac.com/2021/01/should-i-still-use-zabbix-in-aws/ (дата звернення 06.06.2024).

18. Grafana vs. Zabbix [Електронний ресурс]. – Режим доступу: https://www.metricfire.com/blog/grafana-vs-zabbix/ (дата звернення 06.06.2024).

19. Web Application Hosting Best Practices [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/ (дата звернення 06.06.2024).

20. A. D. Tudosi, D. G. Balan and A. D. Potorac, "Secure network architecture based on distributed firewalls," 2022 International Conference on Development and Application Systems (DAS), Suceava, Romania, 2022, pp. 85-90, doi: 10.1109/DAS54948.2022.9786092.

21. What Percentage of Servers Run Linux? [Електронний ресурс]. – Режим доступу: https://frameboxxindore.com/linux/you-asked-what-percentage-of-servers-run-linux.html (дата звернення 06.06.2024).

ДОДАТОК

Скріпти для роботи з БД.

Скріпт 1:

```
-- Створення бази даних
CREATE DATABASE IF NOT EXISTS
my_database;
USE my_database;

-- Створення таблиці
CREATE TABLE IF NOT EXISTS
my_table (    id INT
AUTO_INCREMENT PRIMARY
KEY,
        name
VARCHAR(255),
        age INT,
        gender ENUM('Male',
'Female', 'Other'),  salary
DECIMAL(10, 2),  location
VARCHAR(50) DEFAULT
'Pune'
)
;
-- Вставка тестових даних
INSERT INTO my_table (name, age, gender,
salary)
VALUES
        ('John', 30, 'Male', 50000.00),
        ('Alice', 25, 'Female', 45000.00),
        ('Bob', 28, 'Other', 48000.00);

-- Вибірка всіх даних з таблиці
SELECT * FROM my_table;
```

Нижче скрипт який вставляє тестові дані в таблицю

Скріпт 2

```
        import
```

```
        mysql.co
        nnector
import csv

# З'єднання з MySQL
сервером conn =
mysql.connector.connect(
host="localhost",
user="your_username",
password="your_passwor
d",
        database="your_database"
)
# Створення
курсора cursor =
conn.cursor()

# Вибірка всіх даних з таблиці
cursor.execute('SELECT * FROM
my_table')
data = cursor.fetchall()

# Перевірка та
провалідація даних for row
in data:
        id, name, age, gender,
salary, location = row     if age <=
0 or age > 150:
      print(f"Некоректний вік для
{name}: {age}")    if gender not in
('Male', 'Female', 'Other'):
      print(f"Некоректна стать для
{name}: {gender}")        if salary <= 0 or
salary > 1000000:
      print(f"Некоректна зарплата для {name}: {salary}")

# Закриття підключення до бази даних
conn.close()

# Запис даних у CSV файл with open('my_data.csv', 'w', newline='') as
csvfile:        fieldnames = ['id', 'name', 'age', 'gender', 'salary']     writer =
```

```
csv.DictWriter(csvfile, fieldnames=fieldnames)      writer.writeheader()
        for row in data:      writer.writerow({'id': row[0], 'name': row[1],
'age': row[2], 'gender': row[3], 'salary': row[4]})

print("Дані перевірено та записано у файл my_data.csv")
```

Скріпт 3 *def validate_data(data):*

```
                specific_string =
        "Некоректна зарплата
        для"    specific_string2=
        “Некоректний вік для”
                specific_string3=
        "Некоректна стать для”
                for row in data:      if
        specific_string in str(row):
        return 0
        return 1

def
read_data_from_file(filena
me):   try:
        with open(filename, 'r') as file:
        lines = file.readlines()
        data = [tuple(map(str.strip, line.split(','))) for line
in lines if line.strip()]               return data   except
FileNotFoundError:
      print(f"File '{filename}' not
found.")      return None

def main():
        filename =
'my_data.csv'        data =
read_data_from_file(filena
me)    if data:
        result = validate_data(data)
print(f"Validation result: {result}") if
__name__ == "__main__":
        main()
```

Скріпт 5

```
import mysql.connector

# З'єднання з MySQL
сервером conn =
mysql.connector.connect(
        host="localhost",
        user="your_username",
        password="your_passwo
rd",
        database="your_database"
)
# Створення
курсора cursor =
conn.cursor()

# Вставка даних def insert_data(name, age, gender, salary):     sql
= 'INSERT INTO my_table (name, age, gender, salary) VALUES
(%s, %s, %s, %s)'  val = (name, age, gender, salary)
        cursor.execute(sql, val)
        conn.commit()

# Приклад вставки даних
insert_data('Michael', 32, 'Male', 55000.00)
insert_data('Emily', 28, 'Female', 48000.00)
insert_data('Alex', 25, 'Other', 50000.00)

# Закриття підключення до бази даних
conn.close()
```

Скріпт 5

```
import unittest

# Маємо три функції
# data_storage.py, data_check_add.py,
periodic_check.py

class
TestPythonScripts(unittest.TestCa
se):     def test_data_storage(self):
```

```
        # Тестування скрипта
зберігання даних          result =
data_storage('test_data')
self.assertTrue(result)

        def test_data_check_add(self):
        # Тестування скрипта перевірки та
додавання даних          result =
data_check_add('test_data', 'test_file.txt')
        self.assertTrue(result)

        def test_periodic_check_positive(self):
        # Тестування скрипта, що перевіряє
файл кожні 5 хвилин      result =
periodic_check('test_file.txt')
        self.assertIn('positive', result)

        def test_response_time(self):
        # Тестування часу відгуку скриптів
        start_time = time.time()
        data_storage('test_data')
        data_check_add('test_data'
, 'test_file.txt')
periodic_check('test_file.txt')
end_time = time.time()
        self.assertLess(end_time - start_time, 2)

        def test_reliability(self):
        # Тестування надійності скриптів
        for _ in range(10000):
        self.assertTrue(data_storage('test_data'))
self.assertTrue(data_check_add('test_data',
'test_file.txt'))
self.assertTrue(periodic_check('test_file.txt'))

if __name__ == '__main__':
        unittest.main()
```