УДК 004.054
DOI https://doi.org/10.32782/2663-5941/2024.1.1/44

Тарновецька О.Ю.
Чернівецький національний університет імені Юрія Федьковича

Газдюк К.П.
Чернівецький національний університет імені Юрія Федьковича

Бален С.М.
SoftServe

Дмитращук К.М.
Чернівецький національний університет імені Юрія Федьковича

ДОСЛІДЖЕННЯ ПІДКЛЮЧЕННЯ ІНТЕРНЕТ СИСТЕМИ ДО МОНІТОРИНГУ, ВИКОРИСТОВУЮЧИ СУЧАСНІ DEVOPS ТЕХНОЛОГІЇ

У роботі було проведено аналіз сучасних DevOps технології, зроблено дослідження існуючих систем моніторингу та розроблено власну систему моніторингу. Під час виконання дослідження та удосконалення методів швидкого підключення інтернет-систем до системи моніторингу з використанням сучасних технологій DevOps у розробці програмного забезпечення був проведений аналіз схожих проектів, таких як UptimeRobot, Secom, site24x7. Це дозволило визначити, що ключовими функціями таких систем є можливість швидкого підключення, мінімальний моніторинг системи, цілодобова підтримка та додаткові опції.

Для ефективної реалізації моніторингової системи була виявлена потреба у розширенні спектру доступних метрик, що стосуються стану віртуальної машини, на якій розташована інтернет-система. Для реалізації системи моніторингу було обрано платформу хмарних обчислень AWS, використано інструмент інфраструктури як коду Terraform, інструмент для налаштування систем та розгортання програмного забезпечення Ansible, інструмент моніторингу та оповіщення систем Prometheus, інструмент аналітики та інтерактивної візуалізації Grafana, інструмент автоматизації Jenkins, а також сервіс для збору показників продуктивності системи та програми collectd.

Були розроблені алгоритми для етапів розгортання та налаштування систем, і виявлено необхідні конфігураційні файли. Процес розгортання моніторингової системи було автоматизовано за допомогою інструмента для автоматизації Jenkins. Розроблений алгоритм готовий до промислового використання та може бути використаний для автоматизації підключення моніторингу до інтернет-системи.

У перспективі ця система може бути покращена за допомогою оптимізації через використання Docker контейнера. Крім того, її функціональність може бути розширена шляхом надання можливості підключення до моніторингу різноманітних систем. Це дозволить системі моніторингу бути більш гнучкою та адаптованою до різних вимог і інфраструктурних особливостей, що можуть виникнути в процесі розвитку та експлуатації. Такі розширення і оптимізації сприятимуть підвищенню ефективності та готовності системи до змін у майбутньому.

Ключові слова: *моніторинг, інтернет система, налаштування, хмарні технології, автоматизація підключення.*

Постановка проблеми. При створенні інтернет-системи розробники передбачають певне навантаження та безперервну ефективність роботи [1]. Однак заздалегідь визначити, яке саме буде це навантаження і коли можуть виникнути проблеми, є вкрай важким завданням. У зв'язку з цим часто використовується спеціальні інструменти, які підключають систему до моніторингу та відстежують її стан [2].

Робота присвячена аналізу та дослідженню автоматизації процесу підключення моніторингу до системи за допомогою сучасних технологій DevOps, що здатні значно спростити та прискорити цей процес. Це дозволить використовувати моніторинг для більшої кількості систем із меншими витратами. У свою чергу, система моніторингу забезпечить швидку реакцію на можливі проблеми до того, як вони стануть очевидними для користувача. Вона також дозволить контролювати потребу у підвищенні потужності інтернет-систем та вивчати можливі проблеми в системах.

Метою роботи є дослідження та вдосконалення способів надання можливості швидкого підключення інтернет системи до моніторингу використовуючи сучасні DevOps технології в розробці ПЗ.

Встановлена мета обумовлює наступні завдання: дослідити існуючі системи моніторингу та доступні технології для створення моніторингу; обрати технології що будуть використовуватись для системи; визначити необхідні для моніторингу метрики [3]; розробити методології підключення системи; описати процес інтеграції та впровадження розроблених методів.

Об'єктом дослідження є технології, які дозволяють здійснювати моніторинг систем.

Предметом дослідження є використання хмарних технологій та програмного забезпечення для розгортання та підключення інтернет систем до моніторингових систем.

Виклад основного матеріалу

Аналіз напрямків використання та аналогів моніторингових систем

Метою використання моніторингових систем є:

1. Отримання сповіщення про те, що веб-сайт перейшов у режим офлайн, важливо для того, щоб бути в курсі подій, що можуть призвести до недоступності сайту. Існує ряд причин, таких як проблеми з хостингом, несподівані зростання трафіку, вразливість перед шкідливим програмним забезпеченням або несправність плагінів, які можуть викликати виходження з ладу. Отже, отримавши таке сповіщення, ви можете швидко втрутитися та вжити заходів для відновлення роботи сайту в інтернеті, мінімізуючи можливі фінансові збитки.

2. Відслідковування стабільності сайту у зазначений період часу. Якщо є розуміння, що саме та коли несправно працює та відома можлива причина виникнення проблем, можна розробити відповідне рішення та гарантувати, що веб-сайт залишатиметься доступним в мережі якнайбільше часу.

3. Покращення ефективності веб-сайту [4], оскільки є можливість реагувати на проблеми негайно, що дозволяє уникнути отримання неприємних повідомлень від користувачів чи клієнтів, адже заходи приймаються при їх виникненні.

Підключення інтернет-системи до моніторингу є важливим та раціональним рішенням, оскільки воно забезпечує отримання інформації щодо стану роботи конкретної системи та можливість отримання сповіщень у випадку виникнення проблем. Такий підхід дозволяє вчасно реагувати на проблеми до того, як вони вплинуть на користувачів, а також тримати своїх клієнтів проінформованими про останні зміни. Це економить час і надає спокій. Більше того, можна отримати цінні дані для вирішення будь-яких проблем та планування стратегії вирішення перебоїв. На ринку моніторингових систем доступний широкий вибір інструментів та технологій, які сприяють вирішенню завдань, пов'язаних з моніторингом.

Для дослідження аналогів було обрано доступні програми [5–7], що наведені в табл. 1.

Таблиця 1

Перелік аналогів системи моніторингу інтернет системи

Назва	Розробник	Умови ліцензії
https://uptimerobot.com/	UptimeRobot	$15 в місяць
https://secom.com.ua/	Secom	Від $10 до $20 + доплата за додаткові функції в місяць
https://www.site24x7.com/	Zoho	Від $9 до $449 в місяць

Опишемо кожну із розглянутих програмних систем.

UptimeRobot має численні переваги, включаючи здатність швидкого підключення, надійну цілодобову підтримку та різноманітні способи отримання повідомлень, такі як електронна пошта, SMS, телефонні дзвінки, Twitter, Slack, Zapier, Telegram, Webhooks, Discord. Додатково, вона забезпечує можливість перевірки SSL сертифікатів, що підвищує рівень безпеки. Проте важливо відзначити, що існують певні недоліки, зокрема, обмежена доступність інформації про інтернет-систему може ускладнити виявлення проблем та прийняття рішень.

Перевагами **Site24x7** є великий вибір послуг, а саме: моніторинг сайту, швидкість веб-сторінки (браузер), моніторинг DNS-сервера, 110+ глобальних локацій моніторингу, моніторинг доступності веб-сайту (HTTP / HTTPS), моніторинг пошкодження веб-сайту, моніторинг передачі FTP, моніторинг REST API, моніторинг сертифікатів SSL/TLS, моніторинг веб-транзакцій (браузер), моніторинг реальних користувачів (веб), моніторинг роботи кінцевих користувачів, контролюйте свою внутрішню мережу, моніторинг доставки пошти, моніторинг веб-служби SOAP, моніторинг сервера Ping, порту (користувацький протокол), сервера POP, сервера IMAP і сервера SMTP.

Secom надає можливість швидкого підключення, цілодобова підтримка гнучкі умови,

можливість встановлення SSL, підтримка WHMCS, перенесення сайтів і даних, налаштування KVS, аудит сайту, аудит серверів. Недоліками програми є мала кількість доступної інформації про інтернет систему.

Аналізуючи перелік аналогів, можна зробити висновок, що більшість з них обмежуються базовим рівнем інформації про стан віртуальної машини, на якій розміщена інтернет-система. Брак деталізації у забезпеченні метрик та параметрів віртуальної машини може обмежувати користувача у вивченні та ефективному вирішенні можливих проблем.

Отже, для розробленої системи моніторингу важливими характеристиками обрано:

1. Швидкість підключення системи: Миттєвий доступ та спроможність оперативно реагувати на будь-які події або відхилення у роботі інтернет-системи.

2. Широкий спектр доступних метрик про стан віртуальної машини: Можливість отримання докладних та розширених метрик, що дозволяють глибоко аналізувати та відслідковувати різні аспекти роботи віртуальної машини, такі як ресурси, навантаження, мережева активність та інші параметри.

Ці характеристики допоможуть не тільки ефективно моніторити роботу інтернет-системи, але й нададуть користувачеві повний контроль та розуміння стану віртуальної машини, що є ключовим для вчасного виявлення, аналізу та усунення будь-яких можливих неполадок чи проблем.

Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для реалізації поставленого завдання, основними етапами є:

1. Створення акаунту на AWS.

2. Написати Terraform plan для двох віртуальних машин.

2.1. Terraform plan для Інтернет системи:

2.2. Terraform plan для моніторингу.

3. Написати Ansible Playbook:

3.1. Ansible Playbook для налаштування віртуальної машини під роботу з Apache Maven та встановлення collectd.

3.2. Ansible Playbook для налаштування віртуальної машини під роботу з Prometheus та Grafana.

3.3. Ansible Playbook для підключення collectd до Prometheus.

4. Вивести метрики на панель.

5. Автоматизувати процес за допомогою Jenkins.

6. Протестувати систему.

Результатом реалізації поставленого завдання є розроблений план по реалізації CI pipeline (конвеєр Continuous Integration) [8] який розгортає інтернет сторінку та її моніторинг у AWS та надає можливість слідкувати за інтернет сторінкою, її станом та ресурсами в реальному часі для своєчасного реагування на проблеми.

Вибір платформи хмарних обчислень AWS для створення системи є раціональним рішенням [9], оскільки AWS надає широкий спектр послуг та інфраструктури для ефективного розгортання та управління системою. При цьому також розглядались найбільш популярні альтернативи AWS, такі як Microsoft Azure, Google Cloud Platform, IBM Cloud, Alibaba Cloud, Oracle Cloud, DigitalOcean, встановлено їхні переваги та недоліки.

Використання Terraform для створення віртуальних машин на AWS дозволяє декларативно визначити інфраструктуру, що полегшує її керування та масштабування. Terraform є інструментом інфраструктури як коду (IaC), що надає можливість створення, зміни та версіонування інфраструктури безпечно та ефективно. Від низькорівневих компонентів, таких як обчислювальні екземпляри, сховище та мережа, до високорівневих компонентів, таких як записи DNS та функції SaaS.

Ansible в свою чергу є потужним інструментом для налаштовування цих віртуальних машин, забезпечуючи консистентність та автоматизацію завдань адміністрування. Ansible призначений для налаштування систем, розгортання програмного забезпечення та виконання складних IT-завдань, таких як безперервне розгортання або безперервні оновлення.

Зібрання метрик за допомогою Prometheus та collectd дозволить ефективно моніторити роботу системи, отримуючи детальну інформацію про різні параметри та показники. Цей інструмент збирає та зберігає метрики у формі часових рядів, що означає, що дані про метрики фіксуються з позначкою часу, в яку вони були зафіксовані, разом із необов'язковими парами ключ-значення, що називаються мітками.

Використання Grafana для відображення цих метрик створить зручний та інтуїтивно зрозумілий інтерфейс для моніторингу та аналізу стану системи. ***Grafana*** є універсальним інструментом для аналізу та взаємодії з даними. З його допомогою ви можете запитувати, візуалізувати, налаштовувати сповіщення та отримувати зрозумілі показники незалежно від місця їх зберігання. Для коректної роботи необхідно правильно визначити джерело даних. В нашому випадку:

```
# Datasource
grafana_datasource_name: Prometheus
grafana_datasource_type: prometheus
grafana_datasource_url_protocol: http
grafana_datasource_url: "{{ ansible_nodename }}:{{ prometheus_port }}"
grafana_datasource_basicauth: "false"
grafana_datasource_basicauth_user:
grafana_datasource_basicauth_password:
grafana_datasource_default: "true"
```

Вибір інструмента автоматизації процесів Jenkins [10] допомагає підвищити швидкість розробки програмного забезпечення через автоматизацію всіх етапів життєвого циклу розробки. Jenkins інтегрує всі необхідні процеси, такі як збірка, документація, тестування, пакування, стадії, розгортання, статичний аналіз та інші.

Безперервну інтеграцію у Jenkins забезпечують плагіни, що дозволяють інтегрувати різноманітні етапи DevOps. Якщо необхідно інтегрувати конкретний інструмент, достатньо встановити відповідні плагіни, такі як Git, проект Maven 2, Amazon EC2, HTML publisher та інші.

Проектування системи моніторингу

Розгортання віртуальних машин

Процес розгортання віртуальних машин за допомогою Terraform включає в себе кілька етапів:

1. Ініціалізація робочого простору (Workspace Initialization):

Після створення конфігураційного файлу Terraform (зазвичай з розширенням .tf), використовуючи команду terraform init, відбувається ініціалізація робочого простору. Під час цього етапу Terraform завантажує необхідні плагіни та інші ресурси, необхідні для розгортання інфраструктури.

2. Планування (Planning):

Команда terraform plan дозволяє зрозуміти, які зміни буде вносити Terraform у розгорнуту інфраструктуру. Під час планування виводиться список ресурсів, які будуть створені, змінені чи видалені.

3. Застосування (Applying):

Після успішного планування можна використовувати команду terraform apply, яка вносить зміни в інфраструктуру відповідно до зазначеного плану. Terraform буде запитувати підтвердження перед виконанням змін.

4. Перевірка результатів:

Після виконання команди terraform apply можна перевірити стан інфраструктури, використовуючи команду terraform show або переглядаючи консольний вивід. Також рекомендується перевірити стан інфраструктури відповідно до існуючих потреб, наприклад, з'єднання з віртуальною машиною через SSH або перегляд даних управління хмарним провайдером.

5. Знищення (Destroy):

Для видалення створених ресурсів можна використовувати команду terraform destroy. Це може бути корисним, наприклад, після завершення тестування або в разі потреби звільнення ресурсів.

AWS пропонує різні типи інстанцій, кожен з яких призначений для різних завдань та має власні характеристики. Ось кілька основних типів інстанцій, які варто враховувати:

1. General Purpose (загального призначення) – t3, t4g: призначені для загальних завдань, включаючи різноманітні застосування та веб-сервери. Вони забезпечують збалансовані обчислювальні та мережеві можливості.

2. Compute Optimized (оптимізовані під обчислення) – c^5: підходить для обчислювально-інтенсивних завдань, таких як наукові обчислення та обробка даних.

3. Memory Optimized (оптимізовані під пам'ять) – r5: забезпечують великі обсяги оперативної пам'яті та призначені для застосувань, які вимагають великої кількості RAM, наприклад, бази даних та аналіз даних.

4. Storage Optimized (оптимізовані під зберігання) – i3: призначені для високопродуктивних додатків з великим обсягом введення/виведення (I/O), таких як бази даних з великим обсягом даних.

5. Accelerated Computing (прискорені обчислення) – p4, inf1, f1: забезпечують прискорене обчислення за допомогою графічних процесорів (GPU) або програмованих гейтових матриць (FPGA).

При виборі типу інстанції слід враховувати такі фактори:

1. обчислювальні потреби;
2. пам'ять;
3. зберігання;
4. бюджет.

Для нашого завдання підійде General Purpose тип, тож порівняємо їх підтипи:

Для розробленої системи обрано тип віртуальної машини t3.micro в регіоні Стокгольм та операційною системою Ubuntu, 22.04 LTS та створено дві такі віртуальні машини.

Для налаштування роботи тестової інтернет системи було виконано наступні кроки:

1. Встановлено останню версію java.
2. Створено групу tomcat.
3. Створено користувача tomcat.
4. Встановлено Tomcat.

Таблиця 2

Порівняння ресурсів віртуальних машин

Віртуальна машина	Віртуальні ЦП	Кредитів ЦП на годину	Пам'ять (ГіБ)	Сховище	Продуктивність мережі (Гбіт/с)
t3.nano	2	6	0,5	Тільки EBS	До 5
t3.micro	2	12	1	Тільки EBS	До 5
t3.small	2	24	2	Тільки EBS	До 5
t3.medium	2	24	4	Тільки EBS	До 5
t3.large	2	36	8	Тільки EBS	До 5
t3.xlarge	4	96	16	Тільки EBS	До 5
t3.2xlarge	8	192	32	Тільки EBS	До 5

5. Створено символічне посилання на apache-tomcat.

6. Завантажено конфігураційний файл.

7. Завантажено зібраний проєкт.

8. Запущено сервіс Tomcat.

Налаштування віртуальної машини для моніторингу включає декілька етапів.

1. Встановлення Prometheus.

2. Встановлення Grafana.

Для компонування яких необхідним інструментом є Ansible. Основні компоненти Ansible включають в себе:

1. Інвентар (Inventory) – це файл або група файлів, які визначають інформацію про машини, з якими Ansible буде взаємодіяти. Містить групи машин, змінні груп, та додаткові параметри, такі як IP-адреси та інші з'єднання.

2. Playbook – це YAML-файл, який містить набір задач, які Ansible повинен виконати на вказаних машинах. Playbook складається з певної кількості plays, де кожен play визначає конкретну роль і викликає задачі.

3. Задача (Task) – це одна конкретна дія, яку Ansible повинен виконати на машинах. Задачі визначаються для кожного play в playbook і виконують конкретні дії, такі як встановлення пакетів, налаштування служб тощо.

4. Роль (Role) – це організаційний одиниця, яка визначає, які задачі слід виконувати на машинах. Ролі дозволяють структурувати playbooks та використовувати їх у різних проектах.

5. Модулі (Modules) – це виконавчі блоки, які використовуються задачами для виконання конкретних дій на машинах. Ansible постачається з багатьма вбудованими модулями, такими як apt, yum, copy, file, service, тощо.

6. Плагіни (Plugins) дозволяють розширювати можливості Ansible за допомогою власних розширень та використовувати власні модулі, інвентар та інше. Ansible підтримує плагіни для різних аспектів свого функціоналу.

7. Facts автоматично зібрана інформація про систему, така як IP-адреси, характеристики системи, розташування тощо. Ansible facts можна використовувати в playbooks для прийняття рішень на основі поточного стану системи.

Підключення інтернет системи до моніторингу

Для підключення інтернет системи до моніторингу необхідно:

1. Встановити collectd на віртуальну машину з інтернет системою.

2. Скопіювати конфігураційний файл для collectd.

Конфігурація плагіна:

```
LoadPlugin write_prometheus
<Plugin "write_prometheus">
        Port "9103"
</Plugin>
```

Система моніторингу виконує розгортання та конфігурацію інтернет-системи, встановлення сервісу для збирання показників продуктивності системи та програми, а також розгортання та налаштування системи для збирання та відображення показників продуктивності.

Було розглянуто алгоритми для розгортання та налаштування систем, і виявлено необхідні конфігураційні файли. У заключному пункті наведена частина конфігураційного файлу сервісу для збирання показників. Ця частина відповідає за підключення до системи моніторингу та передачу показників.

Автоматизація процесу розгортання моніторингової системи передбачає використання менеджера пакетів для встановлення Jenkins на машині, з якої буде запускатися весь процес. Для цього необхідно завантажити ключ для доступу до облікового запису AWS та GitHub, встановити та налаштувати необхідні плагіни, зокрема:Terraform, Ansible, Maven, AWS credentials, GitHub credentials.

Наступним кроком є н**аписання пайплайнів для Jenkins робіт, а саме** для запуску Terraform та Ansible.

Для доступу до AWS використано наступну команду у pipeline:

withAWS(credentials: ‘aws_key’)

Де ‘aws_key’ – заздалегідь вказані у Jenkins логін та пароль до AWS.

Для підвищення ефективності, після успішного завершення кожної роботи Jenkins наступну роботу слід викликати за допомогою конструкції:

```
post {
    success {
     build job: '*назва наступної роботи'
   }
}
```

Фінальне представлення проєкту зображено на діаграмі (Рис. 1).

Користувач DevOps взаємодіє з інфраструктурою, використовуючи послуги AWS, що включають такі компоненти:

I. DevOps: Користувач, відповідальний за розробку та операції, взаємодія з AWS інфраструктурою.

II. AWS Cluster:

1. Master Cluster: Jenkins – інструмент автоматизації для безперервної інтеграції та доставки, що взаємодіє з Agents Cluster (Terraform) та Nodes Cluster. Terraform – Інструмент для створення та управління інфраструктурою як коду. Взаємодіє з Nodes Cluster.

2. Nodes Cluster: Ubuntu та CentOS – віртуальні машини, які використовуються для різних потреб проекту.

Рис. 1. Діаграма віртуальних машин

На рис. 2 представлено деталізовану структуру проекту.

Розглянемо компоненти та їх взаємодії:

1. Клієнт (Client):

• Роль: Представник зовнішнього середовища, який взаємодіє з системою.

2. Terraform:

• Роль: Інструмент для створення та управління інфраструктурою як коду.

3. AWS Cluster:

• Nodes Cluster:

• Ubuntu Node:

• Ubuntu:

• Роль: Віртуальна машина з операційною системою Ubuntu.

• Aplications Cluster:

• PostgreSQL:

Роль: База даних PostgreSQL.

• CentOS Node:

• CentOS:

• Роль: Віртуальна машина з операційною системою CentOS.

• Aplications Cluster:

• Maven 3.6, npm, Geocitizen, Tomcat 9, Java 1.8:

• Роль: Різні компоненти додатків та середовище для їх запуску.

• Взаємодія: Maven, npm, та інші компоненти використовуються для збірки та розгортання Geocitizen.

• Master Cluster:

• Jenkins:

• Роль: Інструмент для безперервної інтеграції та доставки.

• Agent Cluster:

• Ansible:

• Роль: Інструмент для автоматизації конфігурації та управління системами.

4. Взаємодії та зв’язки:

• Terraform >> Jenkins: Terraform використовується для автоматизованого розгортання інфраструктури, його виведено в Jenkins.

• Geocitizen Website>>PostgreSQL: З’єднання між веб-додатком Geocitizen і базою даних PostgreSQL.

• Geocitizen Website>>Route53>>Client: Маршрутизація домену за допомогою AWS Route53, яка взаємодіє з клієнтом.

Висновки. Під час реалізації дослідження та вдосконалення способів надання можливості швидкого підключення інтернет системи до моні-

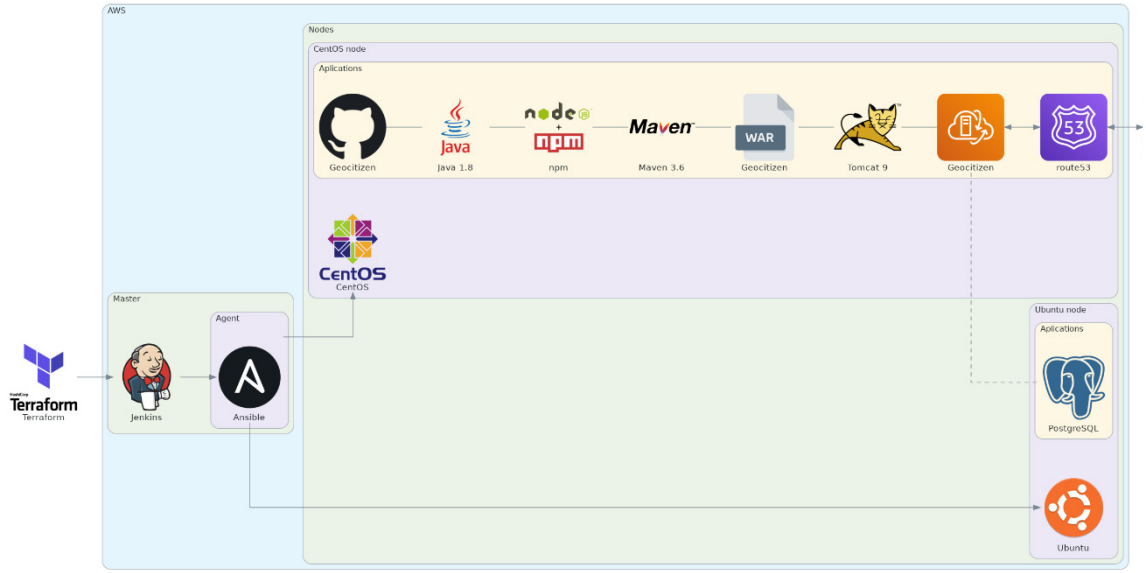

Рис. 2. Діаграма налаштування та роботи

торингу використовуючи сучасні DevOps технології в розробці ПЗ було проведено аналіз аналогічних розробок, таких як UptimeRobot, Secom, site24x7, що дозволило визначити, що основними функціями подібних систем є можливість швидкого підключення, мінімальний моніторинг системи, цілодобова підтримка та додаткові опції. Однак для ефективної реалізації даної моніторингової системи було визначено необхідність розширення спектру доступних метрик, що стосуються стану віртуальної машини, на якій розташована інтернет-система.

Для реалізації системи було обрано платформу хмарних обчислень AWS, інструмент інфраструктури як коду Terraform, інструмент для налаштування систем та розгортання програмного забезпечення Ansible, інструмент моніторингу та оповіщення систем Prometheus, інструмент аналітики та інтерактивної візуалізації Grafana, інструмент автоматизації Jenkins, а також сервіс для збирання показників продуктивності системи та програми collectd.

Розроблено алгоритми для етапів розгортання та налаштування систем, і виявлено необхідні конфігураційні файли. Процес розгортання моніторингової системи було автоматизовано за допомогою інструмента для автоматизації Jenkins. Розроблений алгоритм готовий до промислового використання та може бути використаний для автоматизації підключення моніторингу до інтернет-системи.

У майбутньому ця система може бути вдосконалена через оптимізацію за допомогою Docker контейнера, а також розширена можливістю підключення різноманітних систем до моніторингу.

Список літератури:

1. Кодола Г. М., Волинець Н. С., Сербулова І. В. Автоматизоване тестування веб-додатків з різнорівневою архітектурою. Вісник НТУ «ХПІ» № 5 (1330), серія «Нові рішення в сучасних технологіях», С. 91–100. DOI: https://doi.org/10.20998/2413-4295.2019.05.12
2. Theo Schlossnagle Monitoring in a DevOps world, February. 2018. Communications of the ACM 61(3): P. 58–61. DOI: https://doi.org/10.1145/3168505
3. Sukumar Mandal Site Metrics Study of Koha OPAC through Open Web Analytics and Piwik Tools. Library Philosophy and Practice (e-journal) 2835. July. 2019.
4. Shyamala K., Kalaivani S., Murugan A. A Framework to improve the Web Performance using Reorganization, Optimized Prediction and Prefetching International Journal of Advanced Technology and Engineering Exploration 8(3):2277-3878. September. 2019. DOI: https://doi.org 10.35940/ijrte.C6332.098319
5. Провідна служба моніторингу безвідмовної роботи. URL: https://uptimerobot.com/
6. Комплексні серверні рішення. URL: https://secom.com.ua/
7. Моніторинг на основі AI для сучасних IT. URL: https://www.site24x7.com/
8. Nikhil Singh, Durgesh Patel, Ankit Raj, Shubham, Ms. Sukhmeet Kour CI/CD Pipeline for Web Applications International Journal for Research in Applied Science & Engineering Technology

(IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue V May 2023. DOI: http://dx.doi.org/10.22214/ijraset.2023.52867

9. Faizan Qaisar, Hammad Shahab, Muhammad Iqbal , Hussain Mahmood Sargana, , Muhammad Aqeel, Muhammad Arslan Qayyum Recent Trends in Cloud Computing and IoT Platforms for IT Management and Development. Pakistan Journal of Engineering and Technology. Vol. 6 No. 1 (2023). DOI: http://dx.doi.org/10.51846/vol6iss1pp98-105

10. Kusumadewi R., Adrian R. Performance analysis of devops practice implementation of ci/cd using jenkins Journal of computer science and information technology, Vol 15, No 2 (2023) .pp. 90-95. DOI: http://dx.doi.org/10.18860/mat.v15i2.17091

Tarnovetska O.Yu., Hazdiuk K.P., Balen S.M., Dmytrashchuk K.M. STUDY OF INTERNET SYSTEM CONNECTION TO MONITORING USING MODERN DEVOPS TECHNOLOGIES

In this work, an analysis of modern DevOps technologies was conducted, research on existing monitoring systems was carried out, and a proprietary monitoring system was developed. During the research and improvement of methods for quickly connecting internet systems to the monitoring system using modern DevOps technologies in software development, an analysis of similar projects such as UptimeRobot, Secom, and site24x7 was conducted. This allowed identifying that key features of such systems include quick connectivity, minimal system monitoring, 24/7 support, and additional options.

For the effective implementation of the monitoring system, there was a need to expand the range of available metrics related to the state of the virtual machine hosting the internet system. To implement the monitoring system, the AWS cloud computing platform was chosen, Terraform was used as an infrastructure as code tool, Ansible for system configuration and software deployment, Prometheus for monitoring and alerting, Grafana for analytics and interactive visualization, Jenkins for automation, and collectd for collecting system performance metrics.

Algorithms were developed for deployment and configuration stages, and necessary configuration files were identified. The deployment process of the monitoring system was automated using the Jenkins automation tool. The developed algorithm is ready for industrial use and can be used to automate the connection of monitoring to the internet system.

In the future, this system can be improved through optimization using Docker containers. Additionally, its functionality can be expanded by allowing the connection of various systems to monitoring. This will make the monitoring system more flexible and adaptable to different requirements and infrastructure peculiarities that may arise in the process of development and operation. Such expansions and optimizations will contribute to increasing the efficiency and readiness of the system for future changes.

Key words: *monitoring, Internet system, configuration, cloud technologies, automation of connection.*