

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)

Кафедра _____ Інформаційних управляючих систем _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ другий (магістерський) _____

_____ Дослідження методів і технологій автоматизації моніторингу
інфраструктур для хмарних інформаційних систем _____
(тема)

Виконав:
студент 2 курсу, групи ІУСТм-21-1 _____
_____ Володимир СКЛЯР _____
(власне ім'я, прізвище)

Спеціальність _____ 122 Комп'ютерні _____
_____ науки _____
(код і повна назва спеціальності)


Тип програми _____ освітньо-професійна _____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ Інформаційні управляючі
системи та технології _____
(повна назва освітньої програми)

Керівник _____ проф. каф. ІУС, Володимир САЄНКО _____
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри



(підпис)

_____ Костянтин ПЕТРОВ _____
(власне ім'я, прізвище)

2022 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
Кафедра _____ Інформаційних управляючих систем _____
Рівень вищої освіти _____ другий (магістерський) _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)
Тип програми _____ освітньо-професійна _____
(освітньо-професійна або освітньо-наукова)
Освітня програма _____ Інформаційні управляючі системи та технології _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)



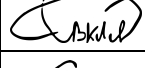
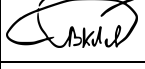

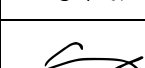
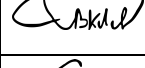


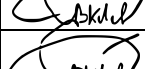
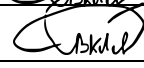
« 21 » листопада 20 22 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____ Скляру Володимирі Олександровичу _____
(прізвище, ім'я, по батькові)

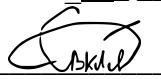
1. Тема роботи _____ Дослідження методів і технологій автоматизації моніторингу інфраструктур для хмарних інформаційних систем _____
затверджена наказом університету від 14 листопада 2022 р. № 1490Ст
2. Термін подання студентом роботи до екзаменаційної комісії 14 12 2022 р.
3. Вихідні дані до роботи технології побудови хмарних інформаційних систем, технології побудови інфраструктур моніторингу, сучасні рішення автоматизації побудови хмарних інфраструктур для систем моніторингу інформаційних систем _____
4. Перелік питань, що потрібно опрацювати в роботі виконати аналіз проблеми автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, провести дослідження технологій побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем, визначити існуючі проблеми та задачі автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, сформулювати рольові вимоги об'єкти автоматизації моніторингу інфраструктур, сформулювати загальні вимоги автоматизації моніторингу інфраструктур, розробити метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, провести апробацію розробленого методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем. _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломне проектування	10.10.2022	
2	Аналіз завдання, літератури та аналогів з теми дипломної роботи	11.10.2022 – 25.10.2022	
3	Постановка задачі	26.10.2022 – 3.11.2022	
4	Аналіз проблеми автоматичного створення хмарних інфраструктур та інфраструктур	4.11.2022 – 10.11.2022	
5	Дослідження методів автоматичного створення хмарних інфраструктур та інфраструктур моніторингу	11.11.2022 – 16.11.2022	
6	Розробка методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем	17.11.2022 – 23.11.2022	
7	Апробація методик автоматизації формування інфраструктур для систем моніторингу хмарних	24.11.2022 – 27.11.2022	
8	Оформлення пояснювальної записки	28.11.2022 – 01.12.2022	
9	Оформлення графічної частини та презентаційних матеріалів захисту	02.12.2022 – 04.12.2022	
10	Представлення на рецензування	05.12.2022	
11	Представлення кваліфікаційної роботи до ЕК	14.12.2022	

Дата видачі завдання 21 листопада 2022 р.

Студент


(підпис)



Керівник роботи

(підпис)

проф. каф.ІУС,Володимир САЄНКО

(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи містить: 96 с., 2 розділи, 42 рис., 10 табл., 21 джерел.

АВТОМАТИЗАЦІЯ, АНАЛІЗ, ІНФРАСТРУКТУРА, КОНФІГУРАЦІЯ, МЕТОДИ, МОНІТОРИНГ, РОЗГОРТАННЯ, СКРИПТ, ХМАРНИЙ.

У роботі виконано огляд методів, моделей, технологій автоматичного створення хмарних інфраструктур та інфраструктур моніторингу. На підставі проведеного аналізу запропоновано методи автоматичного створення хмарних інфраструктур та інфраструктур моніторингу.

Об'єктом дослідження в рамках магістерської кваліфікаційної роботи є інформаційні ресурси хмарних інфраструктур та інфраструктур моніторингу.

Предмет дослідження: моделі та технології автоматичного створення хмарних інфраструктур та інфраструктур моніторингу.

Результати роботи:

- аналіз проблеми автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем;
- дослідження технологій побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем;
- існуючі проблеми та задачі автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем;
- рольові вимоги об'єкти автоматизації моніторингу інфраструктур;
- загальні вимоги автоматизації моніторингу інфраструктур;
- метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем;
- апробація розробленого методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.

ABSTRACT

The explanatory note to the qualification work contains: 96 pages, 2 sections, 42 figures, 10 tables, 21 sources.

ANALYSIS, AUTOMATION, CLOUD, CONFIGURATION, DEPLOYMENT, INFRASTRUCTURE, METHODS, MONITORING, SCRIPT.

The work includes an overview of methods, models, technologies for automatic creation of cloud infrastructures and monitoring infrastructures. Based on the analysis, methods of automatic creation of cloud infrastructures and monitoring infrastructures are proposed.

The object of research within the framework of the master's qualification work is the information resources of cloud infrastructures and monitoring infrastructures.

Subject of research: models and technologies of automatic creation of cloud infrastructures and monitoring infrastructures.

Work results:

- analysis of the problem of automating the formation of infrastructures for monitoring systems of cloud information systems;
- research of technologies for building cloud infrastructures for monitoring systems of cloud information systems;
- existing problems and tasks of automating the formation of infrastructures for monitoring systems of cloud information systems;
- role requirements of infrastructure monitoring automation objects;
- general requirements for automation of infrastructure monitoring;
- a method of automating the formation of infrastructures for monitoring systems of cloud information systems;
- approval of the developed method of automating the formation of infrastructures for monitoring systems of cloud information systems.

ЗМІСТ

Календарний план	5
Скорочення та умовні позначки	8
Вступ.....	9
1 Аналіз проблеми автоматизації моніторингу інфраструктур для хмарних інформаційних систем	11
1.1 Огляд сучасних концепцій автоматизації моніторингу інфраструктур для хмарних інформаційних систем.....	11
1.2 Огляд існуючих рішень побудови автоматизованих хмарних інформаційних систем	13
1.3 Огляд інструментальних засобів побудови хмарних інформаційних систем	14
1.3.1 Огляд інструментальних засобів моніторингу хмарних інформаційних систем	14
1.3.2 Огляд інструментальних засобів хмарних інформаційних систем...	16
1.3.3 Огляд інструментальних засобів впровадження CI/CD процесу для побудови хмарних інформаційних систем	16
1.3.4 Огляд інструментальних засобів для менеджменту конфігурацій хмарних інформаційних систем	17
1.3.5 Огляд інструментальних засобів для написання інфраструктур у вигляді коду для хмарних інформаційних систем.....	18
1.4 Огляд технологій моніторингу хмарних інформаційних систем.....	19
1.5 Висновки з аналізу інструментальних засобів.....	23
1.6 Опис постановки задачі дослідження	24
2 Розробка моделей та методів автоматизацій формування інфраструктур для систем моніторингу хмарних інформаційних систем	26
2.1 Опис об'єкту дослідження	26
2.2 Користувачі системи дослідження	28

2.3 Детальний опис об'єкту дослідження	31
2.4 Функціональні вимоги до інформаційної системи моніторингу	34
2.5 Загальні вимоги автоматизації моніторингу інфраструктур	42
2.6 Метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем	46
2.7 Пояснення до методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем (передумови, формування бібліотеки скриптів).....	48
2.8 Пояснення до методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем (розгортання змін)...	52
2.9 Terraform інфраструктура методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем ...	54
2.10 Ansible інфраструктура методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем ...	58
2.11 Технології реалізації інфраструктури методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем	60
2.12 Технології реалізації моніторингу методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем ...	64
Висновки	68
Перелік джерел посилання	69
Додаток А сертифікати кваліфікації	71
Додаток Б графічний матеріал	72

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;
ВМ – віртуальна машина;
ІС – інформаційна система;
ОС – операційна система;
ALB – Application Load Balancer;
AMI – Amazon Machine Images;
ASG – Auto Scaling Group;
AWS – Amazon Web Services;
CD/ CI – Continuous Delivery/ Continuous Integration;
CLI – Command Line Interface;
CPU – Central Processing Unit;
CW – Cloud Watch;
EC2 – Elastic Compute Cloud;
ELK – Elasticsearch Logstash Kibana;
ER – Entity Relationship;
GCP – Google Cloud Platform;
HTTP – Hypertext Transfer Protocol;
IAM – Identity and Access Management;
ISP/CSP – Internet/Cloud Service Provider;
NAT – Network Address Translation;
SAAS – Software As A Platform;
SG – Security Group;
SSD – Solid State Drive;
SSH – Secure Shell Protocol;
S3 – Simple Storage Service;
UML – Unified Modeling Language;
VPC – Virtual Private Cloud.

ВСТУП

Задачі автоматизації роботи множинних елементів інфраструктури систем зустрічаються в багатьох напрямках та галузях управління архітектурою хмарового середовища. Інформація та її використання стає сьогодні одним із вирішальних факторів у розвитку людства та суспільства. На сьогодні інформаційні системи (ІС) надають усе більше можливостей отримувати та обробляти інформацію та знання, які з нею пов'язані [1].

Для закладів, які використовують ІС існує багато систем розгортання хмарової інфраструктури та інфраструктури моніторингу середовища. Деякі системи поширюються у вільному доступі, деякі розроблюються під кожен компанію або проєкт.

ІС системи автоматичного розгортання інфраструктури отримали великий поштовх до розвитку завдяки розповсюдженому використанню хмарних технологій та розроблених для поліпшення автоматизації практик та інструментів.

Наразі, моніторинг інформаційних систем є невід'ємною складовою будь-якої інфраструктури. Необхідність вчасно та правильно отримувати інформацію про стан інфраструктури, доступності і продуктивності додатків, можливі проблем з безпекою тільки зростає. Створюються та розповсюджуються інструменти, професії та практики акцентовані на вирішення задач, які пов'язані з моніторингом інфраструктури.

Актуальність цієї роботи обумовлена необхідністю забезпечення різноманітних інформаційних систем автоматичними засобами розгортання інфраструктури. До того ж вносити корективи у системи з мінімальними затратами часу і маніпуляціями з конфігурацією.

В ході роботи необхідно провести аналіз предметної області, огляд сучасних концепцій автоматичного створення хмарних інфраструктур та інфраструктур моніторингу, сформулювати проблеми та фактори. Провести

дослідження технологій побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем, сформувані рольові вимоги об'єкту автоматизації моніторингу інфраструктур, загальні вимоги автоматизації моніторингу інфраструктур. Виконати опис постановки задачі дослідження та апробацію розробленого методу.

Предмет дослідження: моделі та технології автоматичного створення хмарних інфраструктур та інфраструктур моніторингу.

Мета проведення досліджень: пошук та розробка шляхів автоматичного створення хмарних інфраструктур та інфраструктур моніторингу.

Наукова новизна: метод автоматичного створення хмарних інфраструктур та інфраструктур моніторингу.

Практична цінність: розроблений метод дозволяє підвищити ефективність автоматичного розгортання хмарної інфраструктури, ефективність автоматичного розгортання інфраструктури моніторингу, ефективності проведення моніторингу.

Перелік задач:

- 1) Виконати аналіз проблеми автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 2) Провести дослідження технологій побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем.
- 3) Визначити існуючі проблеми та задачі автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 4) Сформувані загальні вимоги автоматизації моніторингу інфраструктур.
- 5) Розробити метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 6) Провести апробацію розробленого методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.

1 АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЗАЦІЇ МОНІТОРИНГУ ІНФРАСТРУКТУР ДЛЯ ХМАРНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Огляд сучасних концепцій автоматизації моніторингу інфраструктур для хмарних інформаційних систем

Як об'єкт автоматизації в роботі досліджується процес формування автоматизації побудови комплексу хмарової інфраструктури та інфраструктури моніторингу та його автоматичного реконфігурування.

Виділяють різні створення архітектури: мануальна; скриптова (полуавтоматична) та автоматична.

Оскільки, система розгортається у хмаровій середі, то конфігурація не статична, а динамічно змінюється.

На даний момент у компаніях та проєктах створення хмарової інфраструктури з моніторингом вирішується без впровадження ІС (мануально).

Для вирішення задачі автоматизації побудови комплексу хмарової інфраструктури використовують переважно ручну працю та низькорівневі скрипти для створення та підтримки інфраструктури. Цей підхід до вирішення задачі має ряд суттєвих недоліків [2]:

- відсутність єдиного інформаційного простору в організації;
- можливість помилок, спричинених людським фактором;
- постійна необхідність підтримувати актуальність скриптів при змінах версій залежностей;
- необхідність ручного перенесення даних;
- збільшення часу на модулювання поведінки інфраструктури при внесенні змін;
- відсутність автоматичного запуску тестування продукту;
- збільшення часу на перевірку безпеки інфраструктури;
- виникнення помилок при формуванні звітів.

Розробка і впровадження автоматизації побудови комплексу хмарової інфраструктури та інфраструктури моніторингу дозволить усунути надані недоліки, а саме: створити єдиний інформаційний простір в організації, підвищити швидкість та зменшити кількість помилок при розгортанні та реконфігурації інфраструктури, зменшити кількість помилок при формування звітів [3].

Написання та розгортання хмарних середовищ та моніторингу складається з таких процесів:

- створення бази скриптів, які необхідні для створення хмарного середовища з моніторингом;
- процеси формування документації, для того щоб аналізувати стан середовищ та навчатися роботі з ними;
- процес формування звітів з безпеки, для того щоб знаходити вразливості у системі.

Аналіз наданих процесів показав, що для ефективності їх виконання можна розробити і використовувати автоматизовану задачу, пов'язану з: створенням бази скриптів для конфігурації процесів розгортання інфраструктури, створення хмарової інфраструктури, створення інфраструктури моніторингу; формування документації та звітів з безпеки; розгортанням інфраструктурних змін.

Аналіз бізнес-процесів, що пов'язані з написанням, розгортанням та моніторингом інфраструктури, показав, що основними функціями, які складають задачу, є:

- формування звіту стану інфраструктури;
- написання скриптів процесів створення інфраструктури;
- написання скриптів створення хмарової інфраструктури;
- написання скриптів створення інфраструктури моніторингу;
- формування звіту з безпеки;
- розгортання інфраструктурних змін.

1.2 Огляд існуючих рішень побудови автоматизованих хмарних інформаційних систем

Існує велика кількість рішень для побудови автоматизованих хмарних інформаційних систем. Вони відрізняються інструментами, адже кожен інструмент створений під окрему задачу. ІС відрізняються оточеннями, де їх створюють. Але у глобальному сенсі вони однакові [4]. Є хмарний провайдер, на якому створюються інфраструктура ІС. У провайдера є сервіси, які використовуються при створенні інфраструктури ІС згідно з вимогами. Є CI/CD інструмент, котрий розгортає зміни у інфраструктурі або створює її з нуля. Є інструменти конфігураційного менеджменту та інструменти описання інфраструктури у вигляді коду для автоматизації змін. На рисунку 1.1. представлена типова автоматизована система інформаційної системи.

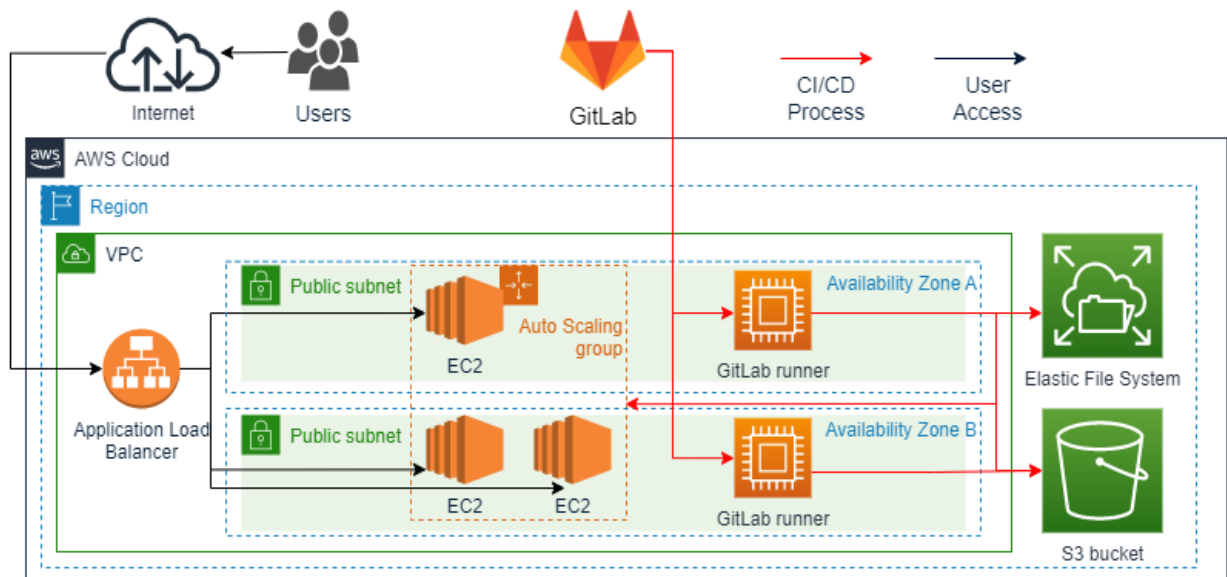


Рисунок 1.1 – Автоматизована хмарна інформаційна система

Компонентами автоматизованої хмарної інформаційної системи на рисунку 1.1 є Application Load Balancer, VPC, Region, Availability Zone, Public Subnet, Auto Scaling Group, EC2, GitLab runner, Elastic File System, S3 bucket.

Application Load Balancer – балансувальник навантаження, котрий працює 7 рівні OSI моделі. Є можливість налаштовувати правила розподілення навантаження. В ньому найбільше можливостей є усіх балансувальник навантаження в AWS.

VPC – віртуальна приватна мережа, в котрій знаходяться віртуальні машини. Мережа складається з підмереж, груп безпеки, таблиць розподілення трафіку, DHCP правил, мережевих ACL.

Region – регіон це набір фізичних датацентрів, які знаходяться поряд, на земній кулі.

Availability Zone – зона доступності це датацентр для зберігання та обчислення даних.

Public Subnet – публічна мережа, яка має доступ до мережі інтернет

Auto Scaling Group – група автоматичного збільшення або зменшення кількості віртуальних машин, в залежності від налаштованих факторів

EC2 – віртуальна машина.

GitLab runner – віртуальна машина із встановленим агентом GitLab, на якій запускається CI/CD процес.

Elastic File System – гнучка файлова система для віртуальних машин.

S3 bucket – сховище об'єктів.

1.3 Огляд інструментальних засобів побудови хмарних інформаційних систем

1.3.1 Огляд інструментальних засобів моніторингу хмарних інформаційних систем

Найбільш розповсюдженими інструментальними засобами моніторингу хмарних інформаційних систем є Prometheus, PagerDuty, AlertManager, Zabbix, ELK, LogStach, Elasticsearch, Kibana, CloudWatch, NewRelic.

Prometheus – система, яка збирає метрики з ендпоінтів та зберігає у себе чи на віддаленому сховищі [5]. Ендпоінти є експортерами, котрі можна писати самому, чи використовувати безліч наявних. Додавати до моніторингу можна все що завгодно. Безкоштовний. Потребує багато ресурсів.

PagerDuty – система оповіщення користувачів про інциденти. Можливо будувати інфраструктури оповіщень [6].

AlertManager – інструмент для оповіщення про інциденти, працює з Prometheus. Найчастіше відправляє повідомлення до PagerDuty. Має можливість сортувати повідомлення, комплектувати їх та вимикати за бажанням.

Zabbix – інструмент для збору та зберігання метрик із серверів, є шаблони для швидкого розгортання моніторингу, є можливість збирати метрики про стан «заліза» серверів. Є можливість створювати оповіщення про інциденти. Для збору метрик необхідно інсталиувати агенти на сервери [7].

ELK – стек розроблений для збирання логів з подальшою обробкою та візуалізацією. Для збирання логів потрібен лог збиральник, який обирається для кожного середовища окремо [8].

LogStash – інструмент агрегації та трансформації логів.

ElasticSearch – база даних для зберігання логів з широкою функціональністю пошуку по логах. Elastic Search проста до масштабування NoSql база даних.

Kibana – засіб для візуалізації логів для кінцевого користувача.

CloudWatch – інструмент для збору та зберігання метрик із інфраструктури AWS. Є можливість налаштувати критичні повідомлення про стан інфраструктури.

NewRelic – інструмент націлений в першу чергу на розробників сервісів. Основна мета інструменту надавати розробникам найбільш детальну та повну інформацію про кожний сервіс, написаний на будь якій мові програмування та розгорнутий на будь якому оточенні. Є можливість створення графіків із метриками та логами та правил надіслання критичних оповіщень [9].

1.3.2 Огляд інструментальних засобів хмарних інформаційних систем

Найбільш розповсюдженими інструментальними засобами хмарних інформаційних систем є AWS, Azure, GCP.

AWS – найбільший хмарний провайдер, найбільше документації, найбільше користувачів, найдовше на ринку, найбільш відмовостійкий [10].

Azure – швидко розвивається, інструмент AzureDevOps стає дуже популярним. Орієнтовний на Microsoft стек.

GCP – найменший з трійки, розвивається повільніше, недостатньо глибокої документації, відділ технічної допомоги не вирішує більшу частину проблем.

1.3.3 Огляд інструментальних засобів впровадження CI/CD процесу для побудови хмарних інформаційних систем

Найбільш розповсюдженими інструментальними засобами впровадження CI/CD процесу для побудови хмарних інформаційних систем є GitLabCI, Jenkins, GitHubCI, AzureDevOps, AWS CodePipeline.

GitLabCI – можливо розгорнути самому, багато документації, проста мова описання пайплайну, зручно використовувати із вбудованим артефакторі та системою контролю версій [11].

Jenkins – можливо розгорнути самому, найбільше можливостей, найскладніша з представлених засобів мова описання пайплайну. Потрібне додаткове рішення для артефакторі та зберігання коду.

GitHubCI – неможливо розгорнути самому, найновіший, найменша кількість можливостей з наявних засобів. Вбудоване артефакторі та система контролю версій. Запускається на агентах, до яких немає доступу.

AzureDevOps – використовується здебільшого з Azure хмарою, замало документації, багато можливостей. Вбудована система контролю версій, артефакторі та менеджер задач.

AWS CodePipeline – використовується з AWS хмарою, не потрібно створювати віртуальні машини для розгортання (дорожче в користуванні), складна система створення інфраструктури CI/CD (складається з окремих складових, котрі потрібно окремо вивчати за налаштовувати).

1.3.4 Огляд інструментальних засобів для менеджменту конфігурацій хмарних інформаційних систем

Найбільш розповсюдженими інструментальними засобами для менеджменту конфігурацій хмарних інформаційних систем Ansible, Chef, Puppet.

Ansible – найлегший для освоювання інструмент розгортання конфігурацій, не потребує головної машини для розгортання змін, не потребує агентів на інших машинах, розгортає зміни за командою. Є вбудований засіб шифрування та дешифрування Anible vault [12].

Chef – тяжкий в освоюванні інструмент, має бекапний сервер, необхідно встановлювати агентів та необхідна головна машина. Агенти самі забирають зміни в конфігурації з головної машини та інсталюють їх за необхідності.

Puppet – тяжкий в освоюванні інструмент, має альтернативний головний сервер, необхідно встановлювати агентів та необхідна головна машина. Агенти самі забирають зміни в конфігурації з головної машини та інсталюють їх за необхідності.

1.3.5 Огляд інструментальних засобів для написання інфраструктур у вигляді коду для хмарних інформаційних систем

Найбільш розповсюдженими інструментальними засобами для написання інфраструктур у вигляді коду для хмарних інформаційних систем є Terraform, CloudFormation, Vagrant.

Terraform – інструмент опису інфраструктури, працює с багатьма провайдерами, має підтримку колективної роботи, зберігає файл із станом інфраструктури, швидко розвивається, найбільше користувачів [13-14]. Має багато вбудованих функцій, котрі з кожним оновленням наближають його до мови програмування. Має можливість будувати граф із залежностями одного ресурсу від іншого – це допомагає використовувати атрибути створеного об'єкту для створення іншого. Приклад Terraform графу представлений та рисунку 1.2. Має можливість планувати та переглядати зміни перед розгортанням інфраструктури. Terraform має власний CLI, котрий розширює функціональність інструменту.

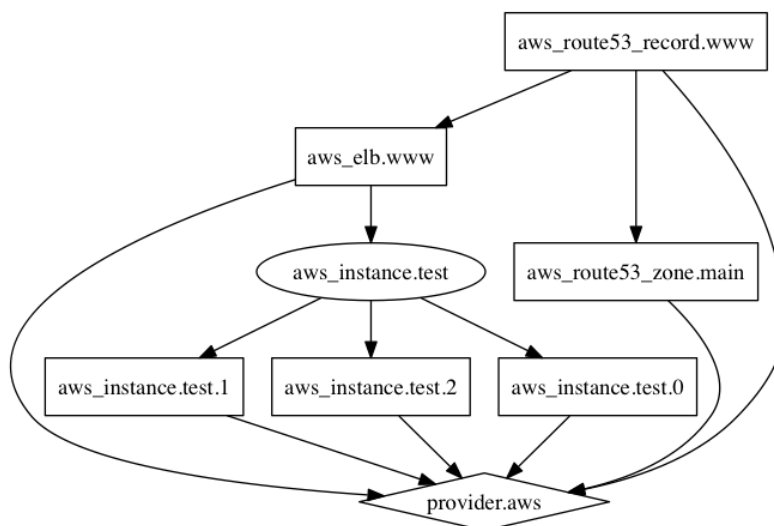


Рисунок 1.2 – Terraform граф

На рисунку 1.2 в Terraform графі розгорнута AWS інфраструктура з трьома віртуальними машинами, балансувальником навантаження, DNS (route53) зоною та записом (DNS ім'я). Балансувальник під'єднується до DNS імені, котре він буде використовувати в інтернеті. Віртуальні машини під'єднуються до балансувальника, через якого будуть отримувати та віддавати трафік.

CloudFormation – специфічний інструмент для розгортання інфраструктури в AWS.

Vagrant – специфічний інструмент для створення віртуальних машин за допомогою Virtual Box.

1.4 Огляд технологій моніторингу хмарних інформаційних систем

Система Prometheus будується навколо Prometheus сервера. Сервер збирає метрики з експортерів. Експортери це програми, які збирають метрики з інших програм та додатків, агрегують їх та надають у вигляді HTTP ендпоінту, яку збирає Prometheus. Зберігає метрики Prometheus у локальному сховищі, також є можливість використовувати інші NoSql БД для зберігання даних. Prometheus може візуалізувати метрики користуючись своїм UI, але для більшої гнучкості використовують Grafana. Прометей виступає джерелом даних для Grafana, використовуючи PromQL для запитів до Prometheus БД. Prometheus має можливість знаходити об'єкти для моніторингу, користуючись пошуком сервісів. Це особливо важливо для роботи з Kubernetes кластерами, де об'єкти створюються динамічно. Prometheus має можливість створювати умови для створення критичних повідомлень про інциденти. Повідомлення надходять до Alertmanager для подальшої обробки та відправлення до інших додатків (PagerDuty, OpsGenie, E-mail та ін.). Приклад Prometheus інфраструктури зазначений на рисунку 1.3.

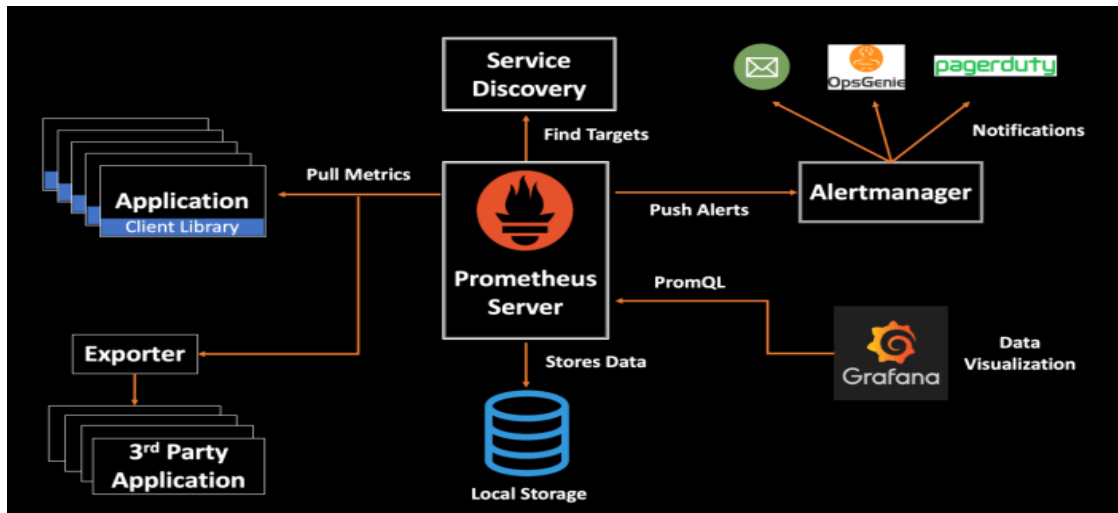


Рисунок 1.3 – Інфраструктура системи Prometheus

Amazon CloudWatch (CW) – це система збирання метрик та логів у AWS. Більшість сервісів AWS (EC2, Lambda, ECS, ECR та ін.) мають можливість надсилати метрики та логи до CW. CW має можливість візуалізувати метрики в інтерфейсі CW чи виступати джерелом даних в Grafana. Базуючись на метриках та логах можливо створювати критичні повідомлення та дії. Критичні повідомлення можливо направляти до SNS сервісу для подальшої відправки до E-mail, PagerDuty, телефонний дзвінок, Slack та інші. Діями можливо автоматизувати роботи системи, наприклад: коли EC2 має проблеми з продуктивністю запуснути ще одну EC2 у ASG. Приклад інфраструктури системи CW зазначений на рисунку 1.4.

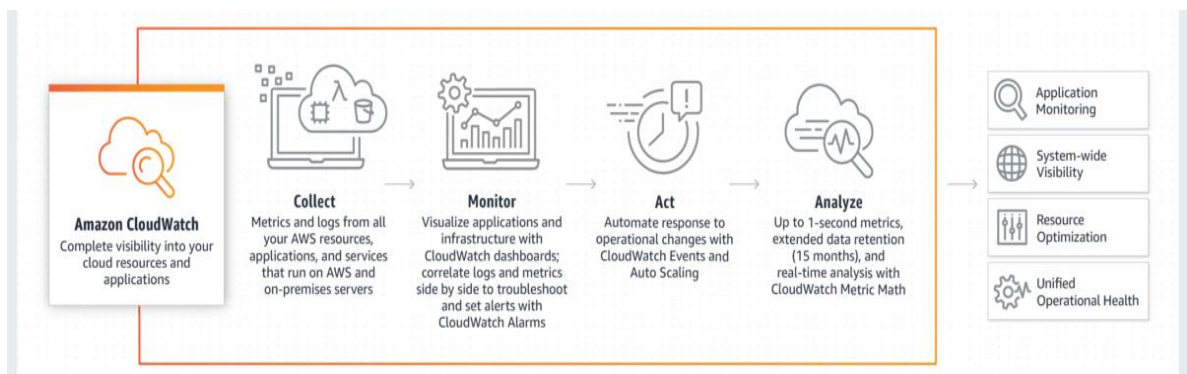


Рисунок 1.4 – Інфраструктура системи CloudWatch

Система ELK створена для роботи з логами. Логи збирається з різних середовищ за допомогою сервісів збору логів. Кожен сервіс створений під свою систему (Linux – filebeat, Windows – winlogbeat та ін.). Сервіси збору логів направляють логи або у LogStash для парсингу, агрегації та фільтрації, або напряму до ElasticSearch. ElasticSearch це NoSQL база даних орієнтовна для сховища логів. База спроектована так, щоб могла зберігати великі різнотипні об'єми логів зі швидким пошуком по ключовим словам. БД має можливість бути невідказною і масштабованою за рахунок системи шардів із даними, які легко масштабувати вшир. Kibana використовується для візуалізації логів за індексами у БД та пошуком використовуючи KibanaQL. Приклад інфраструктури системи ELK зазначено на рисунку 1.5.

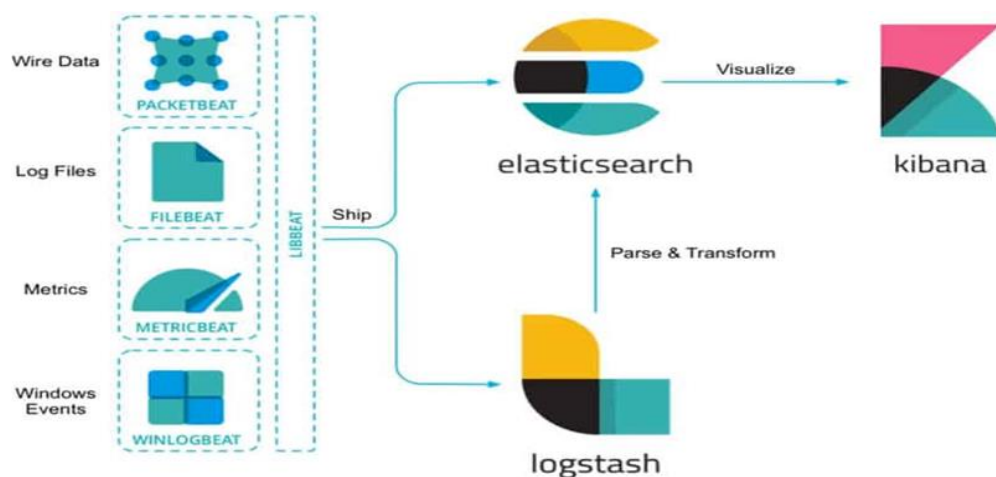


Рисунок 1.5 – Система ЕЛК

Zabbix система дозволяє будувати високостійку інфраструктуру моніторингу різноманітних складових інфраструктури інформаційної системи. Здебільшого Zabbix використовується для моніторингу віртуальних машин, які знаходяться у власних датацентрах. Це стає можливим завдяки IPMI протоколу збору даних з обладнання складових інфраструктури. Користувачі спостерігають за станом інфраструктури використовуючи web UI Zabbix – Zabbix Web Interface. Який візуалізує метрики та надає усю необхідну

інформацію про інфраструктуру системи моніторингу. У центрі Zabbix системи виступає Zabbix сервер, який керує всією інфраструктурою Zabbix системи. Метрики зберігаються на Zabbix БД. Для збору метрик з віртуальних машин використовуються Zabbix агенти, які на ці машини повинні бути встановлені. Zabbix має можливість використовувати HashiCorp Vault для шифрування метрик у системі та зберігання їх в зашифрованому вигляді. Для побудови розподіленої інфраструктури моніторингу використовуються Zabbix проксі-сервери, які дозволяють регулювати навантаження та організовувати моніторинг розподіленої інфраструктури. Для високої стійкості всі елементи Zabbix інфраструктури можуть мати додаткові копії. Для створення інфраструктур моніторингу Zabbix власники та широка спільнота користувачів створюють шаблони, які можливо імпортувати в Zabbix. Zabbix має можливість знаходити віртуальні машини для моніторингу за допомогою правил виявлення. За допомогою графів можливо візуалізувати метрики або використовувати Zabbix як джерело даних в Grafana. У Zabbix можливо створювати правила критичних інцидентів та відправляти їх до необхідних систем оповіщення (PagerDuty, E-mail та ін.) або виконувати дії на віртуальних машинах. Приклад інфраструктури системи Zabbix зазначено на рисунку 1.6.

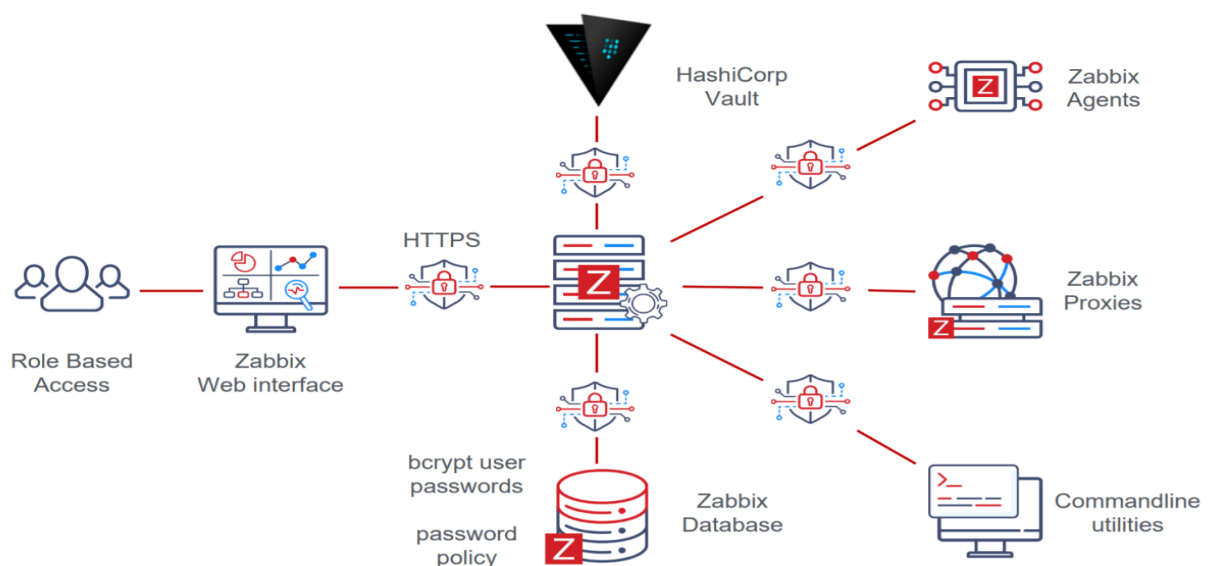


Рисунок 1.6 – Система Zabbix

1.5 Висновки з аналізу інструментальних засобів

В результаті проведеного аналізу були виявлені рекомендації щодо обрання компонентів структури системи моніторингу та інструментальних засобів, які дозволяють побудувати автоматизовану систему створення інфраструктури моніторингу. Рекомендації надані у таблиці 1.1.

Таблиця 1.1. – Рекомендації інструментальних засобів

Галузь	Інструмент	Переваги
Засіб моніторингу	Prometheus	не є складним в ініціалізації, із ним рекомендується використовувати експортери для моніторингу стану контейнерів та віртуальних машини.
Оповіщення про інцидент	AlertManager та PagerDuty	Дозволяють будувати інфраструктури оповіщень та групувати їх за критеріями
Хмарний провайдер	AWS	Найбільш вімовостійкий інструмент із найповнішою документацією
CI/CD засіб	GitLabCI	Має усі необхідні вбудовані можливості (артефакторі, сховище коду)
Конфігураційний менеджмент	Ansible	Простий інструмент для конфігурації та використання
Інфраструктура як код	Terraform	Terraform інструмент розгортає інфраструктуру на різних платформах. Наявна функціональність для будування технологій автоматизацій за принципами CI/CD.

1.6 Опис постановки задачі дослідження

Автоматизацію моніторингу у загальному випадку можливо розбити на задачі. Задача формування інфраструктури для вирішення задач моніторингу в хмаровому середовищі для створення інфраструктури. Задача формування раціональної структури системи моніторингу хмарових систем для покращення вимірів моніторингу та покриття необхідних частин системи моніторингом. Задача оптимізації процесу моніторингу хмарової системи для зменшення грошових та часових затрат, які витрачається на процес моніторингу. Рішення задач автоматизації моніторингу представлено на рисунку 1.7.

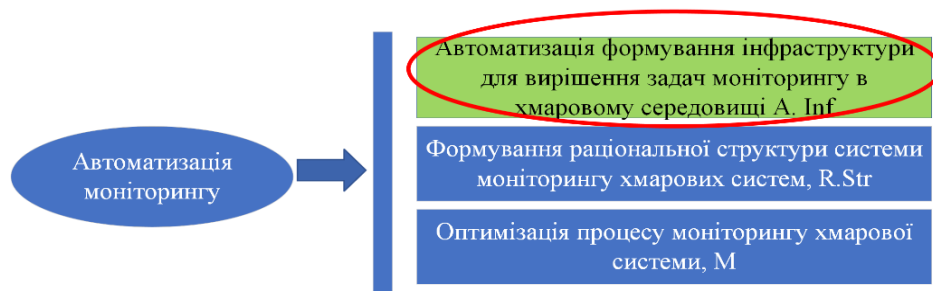


Рисунок 1.7 – Обрана задача автоматизації

На рисунку 1.7 обрана задача автоматизації формування інфраструктури для вирішення задач моніторингу в хмаровому середовищі, на основі якої буде створюватися метод.

Основним критерієм системи $\epsilon J_{Inf} \rightarrow optimum$.

Базуючись на рекомендаціях інструментальних засобів, як хмарне оточення виступає AWS.

Обмеження: хмарне оточення AWS.

Вихідним результатом є метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Доступ до даних буде здійснюватися, використовуючи декілька БД, а саме:

- база даних для збереження користувачів задачі, для логіну у різні частини системи;
- бази даних для засобів моніторингу, які будуть зберігати метрики;
- база даних для налагодження процесу доставки інциденту до користувача (команди, політики ескалації, сервіси).

Інформаційний комплекс повинен бути спроектований з урахуванням можливості збільшення кількості і зміни складу і змісту технологічних процедур предметної області.

Для роботи з системою, необхідний персонал, який має глобальне розуміння роботи хмарних середовищ.

Пошук даних в ІС, виконання запиту повинні займати не більше 5 хвилин, зберігання даних повинне займати не більше 1 хвилини, формування звітів – не більше 10 хвилин.

Для коректності і правильності вводу даних необхідний контроль вводу даних, наявність повідомлень, що попереджують про помилки, сторінки підтвердження.

ІС повинна допускати нарощування функціональних можливостей на основі використання БД.

ІС повинна бути високостійка та готова до горизонтального та вертикального масштабування.

ІС повинна забезпечувати можливість одночасної роботи декільком користувачам і забезпечити захист від несанкціонованого доступу шляхом авторизації користувача під своїм логіном. При цьому виконується розмежування доступу користувачів за ролями для різних типів задач.

2 РОЗРОБКА МОДЕЛЕЙ ТА МЕТОДІВ АВТОМАТИЗАЦІЙ ФОРМУВАННЯ ІНФРАСТРУКТУР ДЛЯ СИСТЕМ МОНІТОРИНГУ ХМАРНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Опис об'єкту дослідження

Об'єкт дослідження є хмарна інформаційна система. Структура хмарної інформаційної системи зазначена на рисунку 2.1.



Рисунок 2.1 – Структурна схема об'єктів дослідження

На рисунку 2.1 структура системи моніторингу необхідна для створення системи моніторингу та проектування хмарної інфраструктури системи моніторингу. Хмарна інфраструктура системи моніторингу необхідна для створення системи моніторингу. Інформаційна система базується на системі моніторингу.

В дослідженні потрібно реалізувати процес моніторингу (рисунок 2.2). Процес моніторингу складається з 3 компонент. Процедура формування інфраструктури для вирішення задач моніторингу в хмарному середовищі для створення інфраструктури. Формування структури системи моніторингу хмарових систем для створення схем і принципів моніторингу. Процес моніторингу хмарової системи для отримання метрик, логів та критичних повідомлень базуючись на стані інфраструктури.

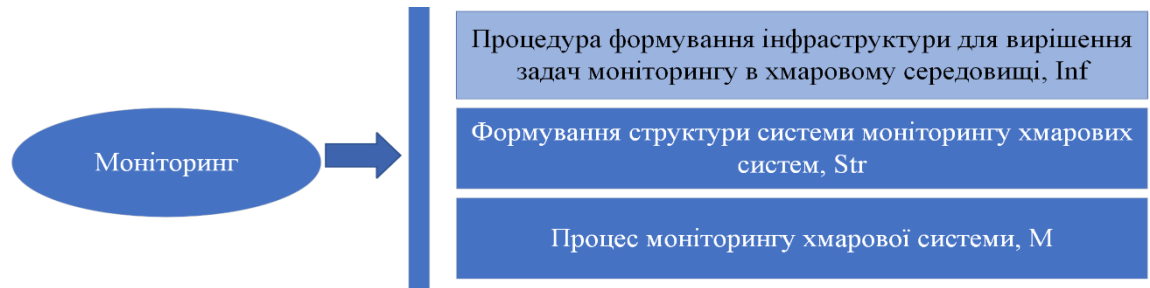


Рисунок 2.2 – Схема процесів для реалізації моніторингу

В дослідженні потрібно реалізувати процес автоматизації моніторингу (рисунок 2.3), який був отриманий із опису постановки задачі дослідження.



Рисунок 2.3 – Схема автоматизації моніторингу

Критеріями системи можуть бути обрані критерії J_{Inf} , J_{Str} , J_M .

$J_{Inf} \rightarrow optimum$ – скорочення часу на розгортання системи моніторингу, зменшення ймовірності виникнення структурних помилок та швидке їх виправлення.

$J_{Str} \rightarrow optimum$ – збільшення швидкості роботи системи, зменшення вартості системи, збільшення зручності.

$J_M \rightarrow optimum$ – збільшення швидкості виконання операцій процесу моніторингу, скорочення споживаних ресурсів, економія витрат.

Реалізація проекту потребує наступних витрат:

- витрати на обстеження предметної області (вивчення організаційної та функціональної структур, існуючих засобів автоматизації, техніко-економічних показників, електронного документообігу, застосовуваних методів проєктування та ін.);
- витрати на використання хмарних технологій;
- витрат на розробку плану проєкту на ІС та його виконання.

2.2 Користувачі системи дослідження

Користувачами системи є: власники системи (Owner), власники інтернет-сервісів або хмарних сервісів (ISP\CSP), адміністратори (DevOps), користувачі (Users), розробники (Developers). Користувачі поділяються на 3 системи. Користувачі всієї системи (рисунок 2.4), користувачі системи моніторингу (рисунок 2.5) та користувачі системи автоматичного розгортання інфраструктури (рисунок 2.6).

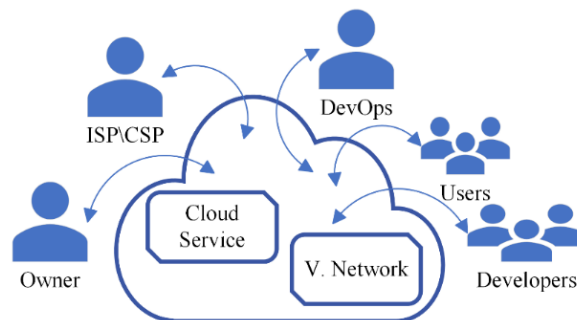


Рисунок 2.4 – Користувачі системи

На рисунку 2.4 система складається з хмарового сервісу (Cloud Service) та віртуальної мережі (V.Network). Хмарний сервіс – хмарний провайдер, який надає сервіси для використання у системі. Віртуальна мережа – мережа, в якій

функціонує система. Власники системи отримують звіти про стан системи та прагнуть до реалізації критеріїв. Власники хмарних сервісів відповідають за оновлення та підтримку сервісів хмарного провайдеру. Адміністратори системи налаштовують процеси створення інфраструктури, моніторингу та їх автоматичного розгортання. Розробники створюють програми та сервіси та хмарному оточені та користуються процесами автоматичного розгортання. Користувачі системи користуються послугами сервісів, створених для них розробниками.

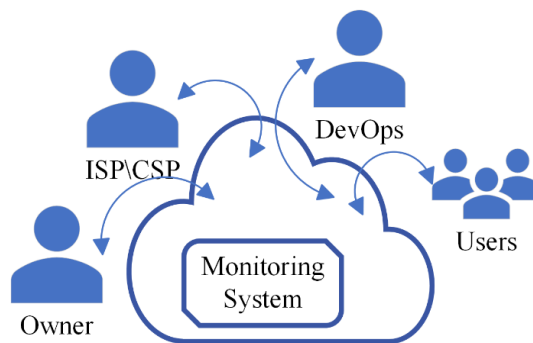


Рисунок 2.5 – Користувачі системи моніторингу

На рисунку 2.5 система є системою моніторингу. Адміністратори в даному випадку поділяються на архітектора створення хмарової інфраструктури та архітектора створення моніторингу інфраструктури. Користувачем такої системи є інженер з моніторингу.

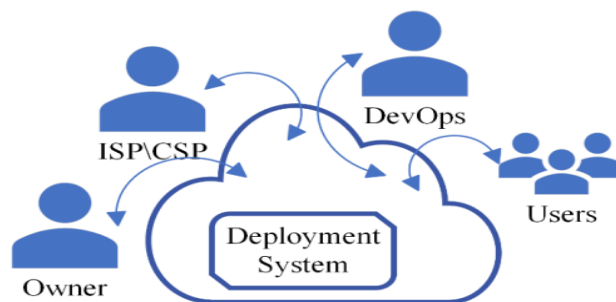


Рисунок 2.6 – Користувачі системи автоматичного розгортання інфраструктури

На рисунку 2.6 система є системою автоматичного розгортання інфраструктури. Адміністратори в даному випадку відповідають за створенням процесів та скриптів автоматичного розгортання інфраструктури. Користувачами такої системи є розробники, котрі використовують підготовлену систему розгортання для розгортання додатків.

Формалізований вигляд для кожного користувача може бути представлений у вигляді виразів:

$$U(\text{Sys}) = \{(\text{Owner}), (\text{ISP}\backslash\text{CSP}), (\text{DevOps}), (\text{Users}), (\text{Developers})\}$$

$$U(\text{infr}) = \{(\text{Owner}), (\text{ISP}\backslash\text{CSP}), (\text{DevOps}), (\text{Users}), (\text{Developers})\}$$

$$U(\text{Str}) = \{(\text{Owner}), (\text{ISP}\backslash\text{CSP}), (\text{DevOps}), (\text{Users}), (\text{Developers})\}$$

$U(\text{M}) = \{(\text{Owner}), (\text{ISP}\backslash\text{CSP}), (\text{DevOps}), (\text{Developers}=\text{User.D}) \text{ розробники (виступають користувачами системи Users.D)}\}$

Основними споживачами результатів моніторингу є власники інформаційної системи (Owner) та адміністратори систем (Admin). Розширена схема взаємозв'язків цих учасників показано на рисунку 2.7.

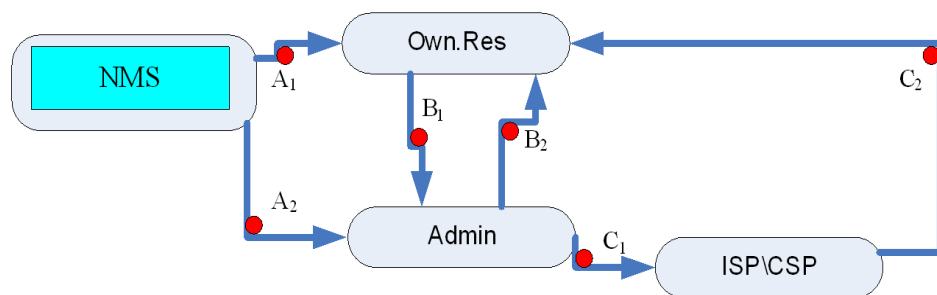


Рисунок 2.7 – Розширена схема взаємозв'язків Owner та Admin учасників

На рисунку 2.7 прийнято позначення:

A1 – інформація про стан системи для Власника (Owner.Res);

A2 – інформація про стан системи для Адміністратора (Admin);

B1 – вимоги для Адміністратора;

B2 – звіт про стан системи для Власника (стан, прогнозування, пропозиції);

C1 – вимоги для провайдерів (ISP\CSP);

C2 – умови договору для Власника (економічні показники).

Результати моніторингу надсилаються окремо Власнику (A1) та Адміністратору (A2). Ці звіти відрізняються тим, що звіт A2 – детальний, а A1 – скорочений. Це лише результати моніторингу без аналітичних висновків.

Адміністратор на основі даних A2 формує звіт для Власника (B2). Власник виходячи з звітів A1, B1 формує вимоги до функціонування системи (B2). Ці вимоги передаються Адміністратору. З вимог B2 Адміністратор формує вимоги до зміни умов формування інфраструктури системи (C1). Ці вимоги надсилаються провайдеру. Провайдер коригує умови договору із Власником (C2).

2.3 Детальний опис об'єкту дослідження

Інформаційна система з одного боку є інформаційним сервісом Service, з іншого боку є програмним продуктом App. Далі позначаємо – Serv.App.

Інформаційну систему як об'єкт контролю можна подати у вигляді чорної скриньки (рисунок 2.8).



Рисунок 2.8 – Схема автоматизованої інформаційної системи

X – вхідні змінні системи, Y – вихідні змінні для системи (QoS), M – конфігураційні змінні для системи.

Інформаційна система функціонує у хмарному середовищі Cloud у межах деякої комп'ютерної мережі. Ця мережа – віртуальна комп'ютерна мережа V.Net.

Віртуальна комп'ютерна мережа V.Net – це віртуальні комп'ютери, об'єднані в комп'ютерну мережу в хмарному просторі Cloud.

Фактично мережа – це комп'ютери із встановленим програмним забезпеченням, що утворюють деяку інфраструктуру – Inf.Net.

Конфігураційні змінні M – це результат функціонування інфраструктури комп'ютерної мережі.

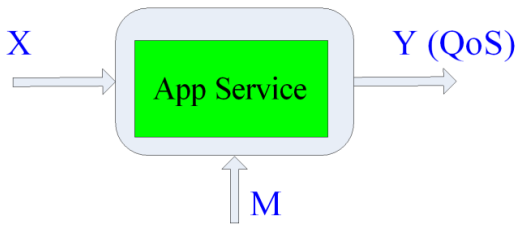
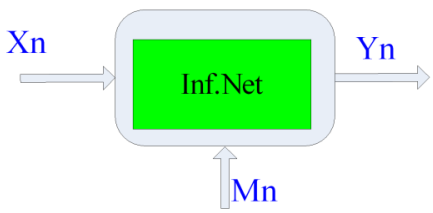
Конфігурація підлаштовується під характеристики якості контрольованої системи Serv.App.

Реалізація процесу моніторингу передбачає можливість реконфігурування системи моніторингу. Є два варіанти:

- автоматичне конфігурування компонентів інфраструктури;
- конфігурування компонентів інфраструктури за допомогою скриптів (напівавтоматичне).

Задача моніторингу інформаційної системи зводиться до двох задач, які зображені у таблиці 2.1.

Таблиця 2.1 – Задачі моніторингу інформаційної системи

Задача 1 – моніторинг самої системи	
Задача 2 – моніторинг інфраструктури для системи	

X_n - вхідні змінні інфраструктури мережі, Y_n - вихідні змінні для інфраструктури мережі, M_n - конфігураційні змінні, вимоги до мережі.

Тоді для завдання моніторингу об'єкт моніторингу розглядається як дві системи, і вирішується обидві завдання (об'єднання Задача 1, Задача 2), як показано на рисунку 2.3.

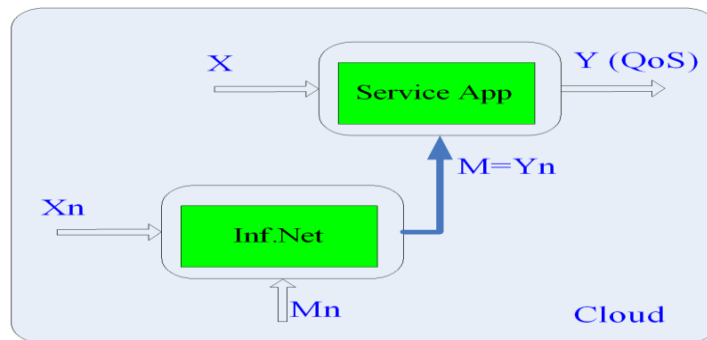


Рисунок 2.9 – Схема завдань моніторингу

Моніторинг реалізується за допомогою додаткових систем – NMS (Network Managemnt System). Схема взаємодії системи моніторингу та об'єкта моніторингу зображена на рисунку 2.10.

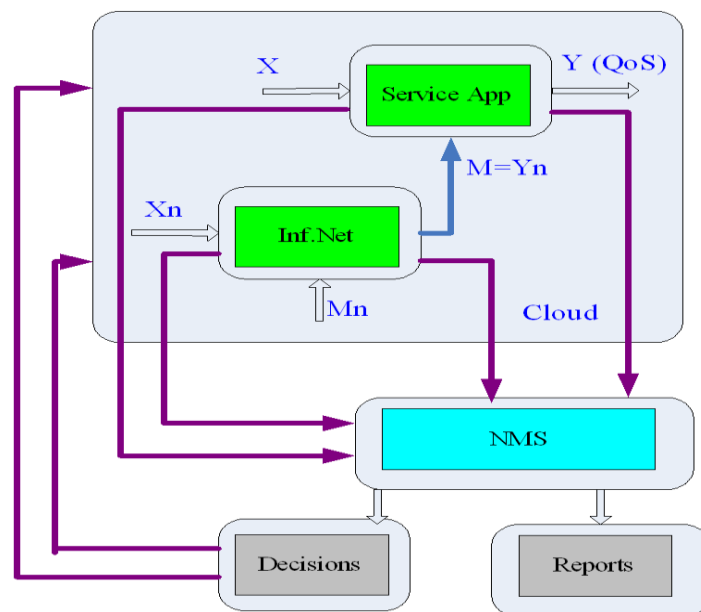


Рисунок 2.10 – Схема взаємодії моніторингу та об'єкту моніторингу

Система моніторингу забезпечує контроль значень вхідних (X , X_n), вихідних (Y , Y_n) та конфігураційних (M , M_n) змінних як самої інформаційної системи, так і її інфраструктури. Система моніторингу пов'язана з блоком прийняття рішень (Decisions) та блоком формування звітів (Reports).

На рисунку 2.10 показано загальну функціональну схему конфігурації системи моніторингу. Конфігурація системи моніторингу для хмарних систем відрізняється від загальних систем. Основна відмінність – багатокомпонентність системи. Це вже не цілісна система з набором інструментальних засобів.

Кожна система вимагає певних ресурсів, спеціальної настройки відповідно до іншими компонентами. Крім того, така структура вимагає спеціального середовища функціонування, яке представляється як інфраструктура комплексу моніторингу.

Таким чином, завдання дослідження зводиться до вирішення питань автоматизації побудови комплексу моніторингу, його автоматизованого реконфігурування та реалізації процесу моніторингу.

2.4 Функціональні вимоги до інформаційної системи моніторингу

Діаграми Use Case, що описують функції були розроблені відповідно до технології побудови діаграм [15].

Use Case контекстна діаграма інформаційної системи моніторингу приведена на рисунку 2.11.

Для опису функціональних вимог до інформаційної системи моніторингу використаємо User stories з переліком необхідних функцій та черговості їх реалізації для кожної функціональної підзадачі визначеній на контекстній Use Case діаграмі.

Сценарії користувача, або User Story – це короткі, прості описи функцій програми, які записуються з точки зору користувача, клієнта системи [16].

Не існує єдиного стандарту формування User story, тому для зручності опишемо User stories у таблиці 2.2.

Відомості про класи інформаційної системи моніторингу містяться в таблиці 2.3.

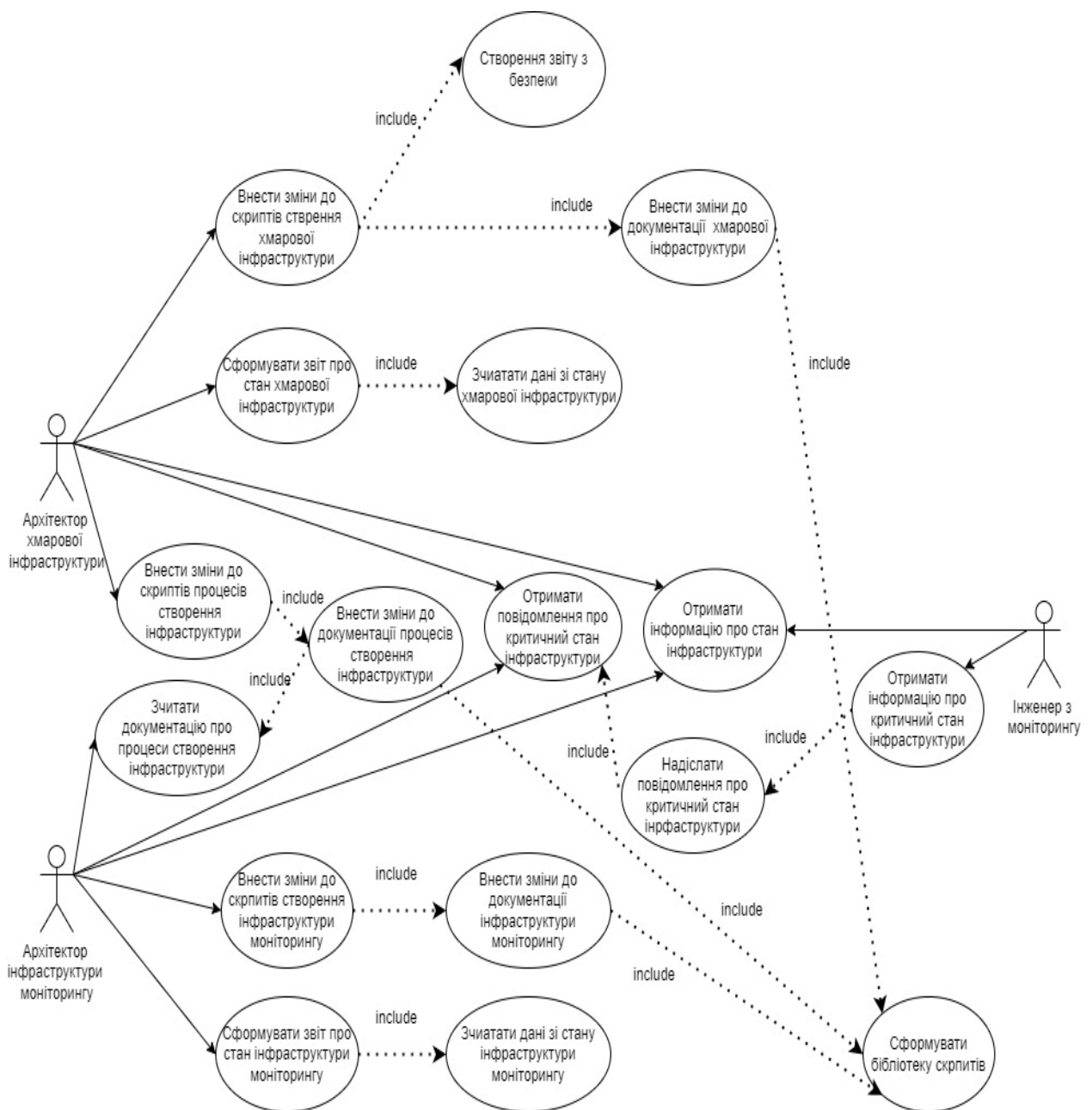


Рисунок 2.11 – Діаграма прецедентів інформаційної системи моніторингу інфраструктури

Таблиця 2.2 – User stories інформаційної системи моніторингу інфраструктури

Номер User story	Опис
1	Архітектору хмарової інфраструктури пропонується створити скрипти процесів створення інфраструктури, щоб на цій основі створювати хмарову інфраструктуру та інфраструктуру моніторингу (рисунок 2.12).
2	Архітектору хмарової інфраструктури пропонується створити скрипти процесів створення хмарової інфраструктури, щоб на цій основі створювати інфраструктуру моніторингу (рисунок 2.13). Умови: реалізована user story 1.
3	Архітектору інфраструктури моніторингу пропонується створити скрипти процесів створення інфраструктури моніторингу, щоб на цій основі отримувати інформацію про стан інфраструктури та кричні повідомлення (рисунок 2.14). Умови: реалізована user story 2.
4	Архітектору хмарової інфраструктури, архітектору інфраструктури моніторингу та інженеру з моніторингу пропонується отримувати інформацію про стан інфраструктури для того, щоб знати стан системи (рисунок 2.15). Умови: реалізована user story 3.
5	Інженеру з моніторингу пропонується отримувати інформацію про кричний стан інфраструктури для того, щоб надіслати інформацію про критичний стан інформації по ескалації (рисунок 2.16). Умови: реалізована user story 3.
6	Архітектору хмарової інфраструктури та архітектору інфраструктури моніторингу пропонується отримувати повідомлення про критичний стан інфраструктури для того, щоб оперативно реагувати на помилки в інфраструктурі (рисунок 2.17). Умови: реалізовані user story 5.

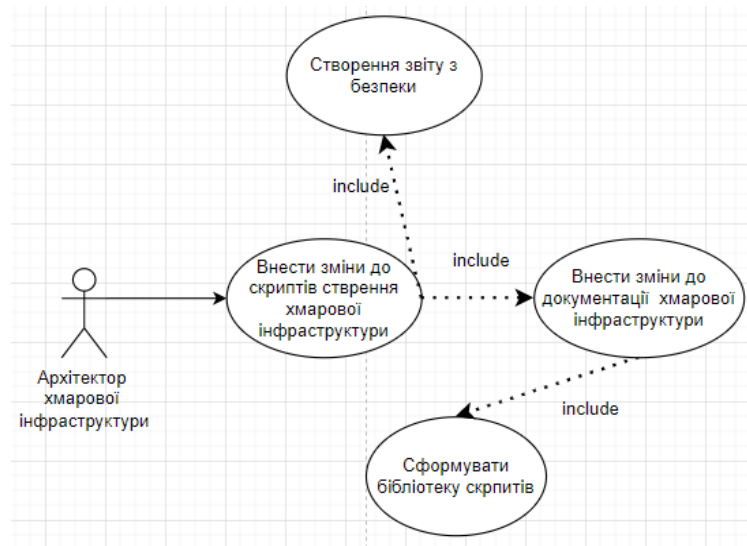


Рисунок 2.12 – Діаграма прецедентів для Use case 1

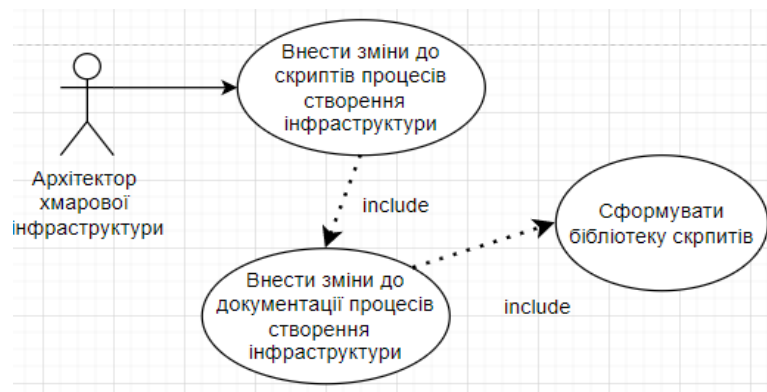


Рисунок 2.13 – Діаграма прецедентів для Use case 2

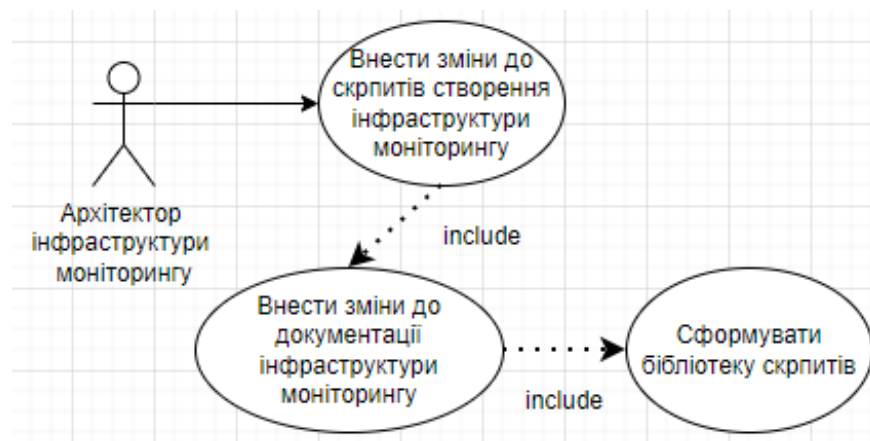


Рисунок 2.14 – Діаграма прецедентів для Use case 3

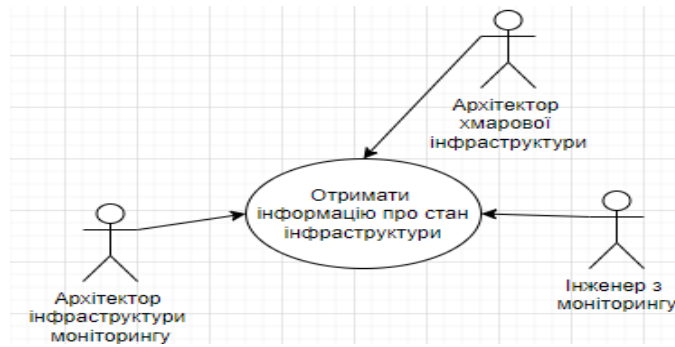


Рисунок 2.15 – Діаграма прецедентів для Use case 4

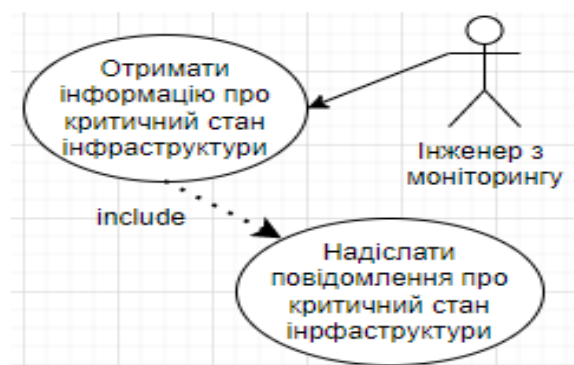


Рисунок 2.16 – Діаграма прецедентів для Use case 5

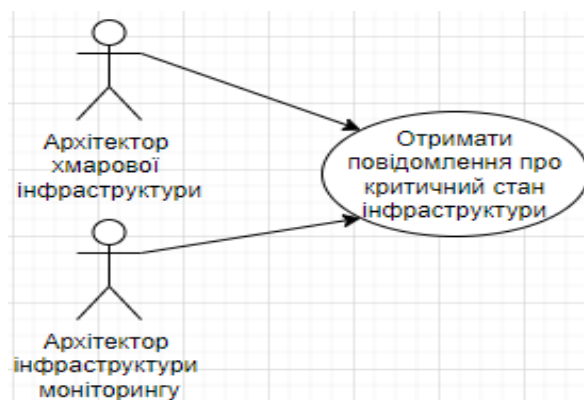


Рисунок 2.17 – Діаграма прецедентів для Use case 6

Діаграма класів, що описує структуру хмарної ІС моніторингу інфраструктур, показуючи класи системи, їх атрибути, операції та взаємозв'язок між об'єктами ІС зображена на рисунку 2.18.

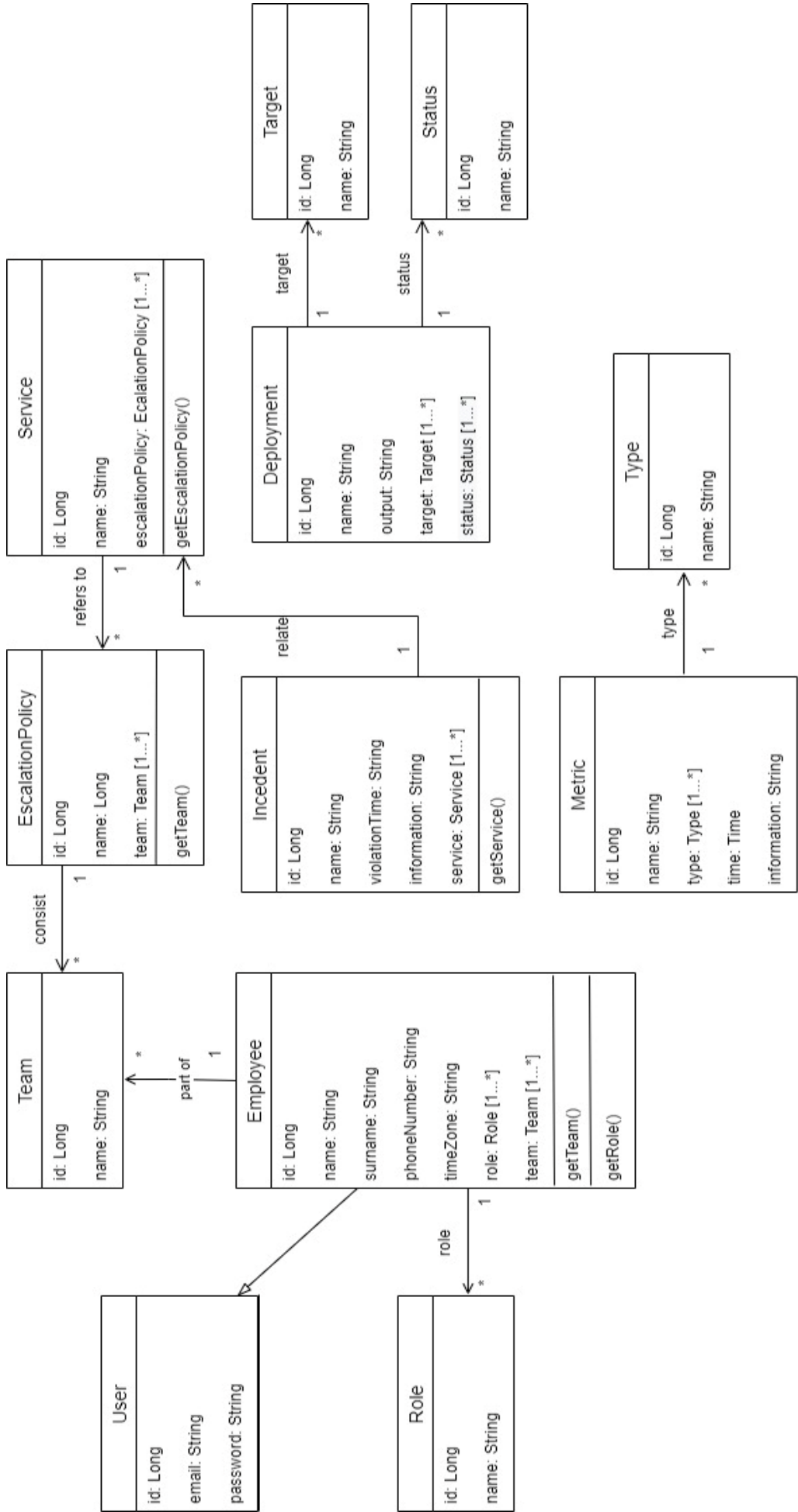


Рисунок 2.18 – Діаграма класів хмарної інформаційної системи моніторингу інфраструктур

Таблиця 2.3 – Опис класів хмарної інформаційної системи моніторингу інфраструктур

Клас	Опис
User	Описує користувача системи
Employee	Описує співробітника
Role	Описує роль співробітника
Team	Описує команду, у якій знаходяться співробітники
EscalationPolicy	Описує політику ескалації для кожної команди (кому і у яку чергу будуть надходити інциденти)
Service	Описує сервіс, який інтегрований з якоюсь системою
Incident	Описує інцидент, який надходить до сервісу
Deployment	Описує основну інформацію про розгортання змін на середовищі
Target	Описує тип середовища для розгортання змін
Status	Описує статус завершення розгортання
Metric	Описує одиницю моніторингу, яка відповідає за надання необхідної інформації
Type	Описує тип для необхідної метрики

Опис концептуальної моделі даних, що зберігаються, хмарної інформаційної системи моніторингу інфраструктур розроблено у вигляді діаграми «сутність-зв'язок» (ER-діаграми). ER-діаграма зображена на рисунку 2.19.

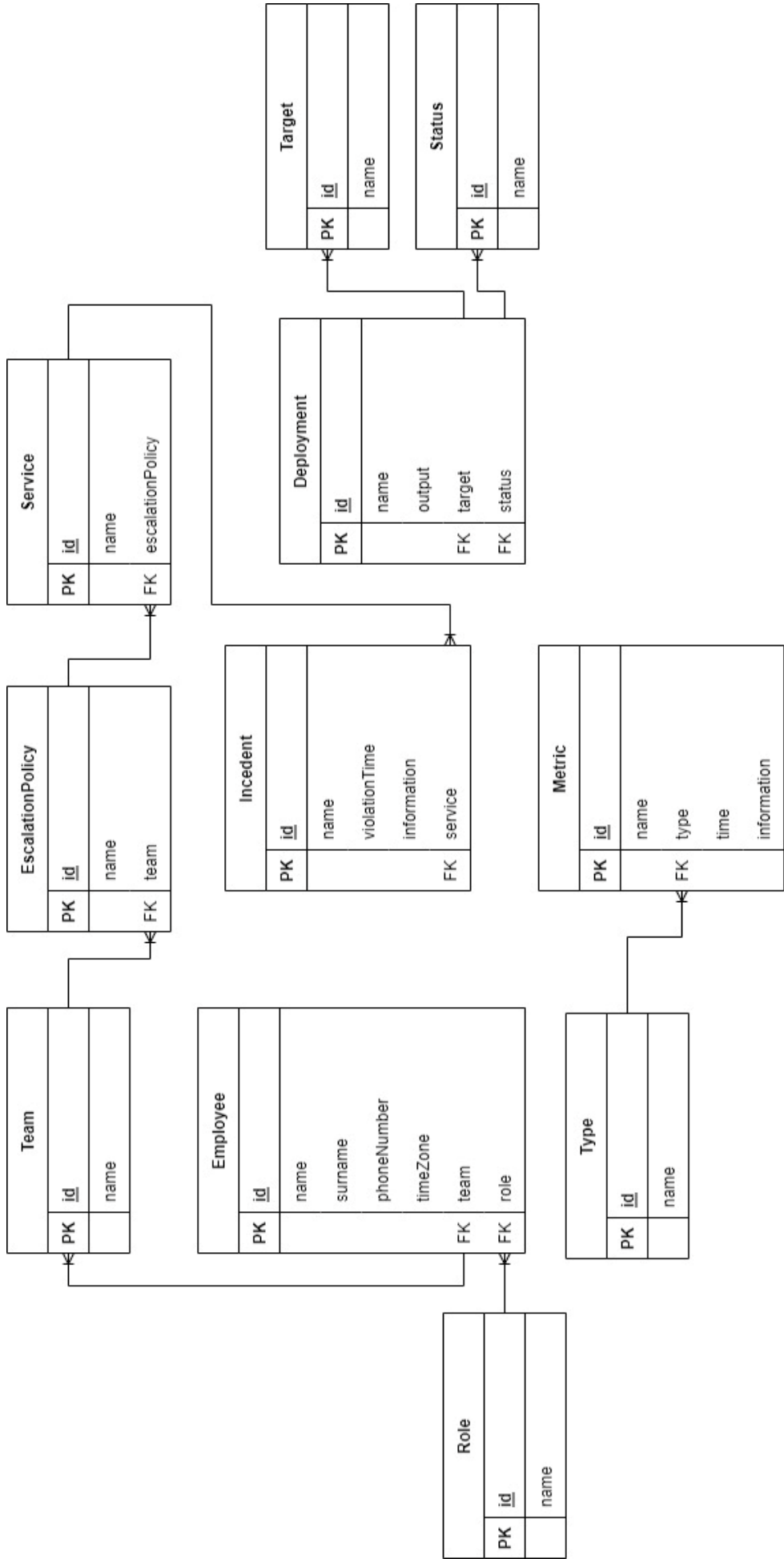


Рисунок 2.19 – Логічна схема даних хмарної інформаційної системи моніторингу інфраструктур

2.5 Загальні вимоги автоматизації моніторингу інфраструктур

Загальні вимоги автоматизації моніторингу інфраструктур складаються з передумови, скрипту gitlab-ci, скрипту Ansible, скрипту Terraform, розгортання Terraform, розгортання Ansible.

1. Передумови використання технологій автоматизації, створення та налаштування аккаунтів, налаштування віртуальних машинами $R1 = \{ \langle \text{AWS аккаунт} \rangle, \langle \text{GitLab сервер} \rangle, \langle \text{PagerDuty аккаунт} \rangle \}$.

AWS аккаунт – авторизований AWS аккаунт для входу до платформи AWS.

GitLab сервер – підготовлений GitLab сервер із налаштованими віртуальними машинами для запуску CI/CD процесу.

PagerDuty аккаунт – авторизований PagerDuty аккаунт для входу до платформи PagerDuty.

2. Скрипт gitlab-ci [17]. $R2 = \{ V_{g1}, V_{g2}, V_{g3}, V_{g4}, V_{g5}, V_{g6} \}$

V_{g1} – оголошення змінних оточення CI/CD процесу для використання при конфігурації CI/CD та надання до змінних оточення Terraform (V_t).

V_{g2} – змінення етапів CI/CD процесу для визначення порядку виконання етапів.

$V_{g3} = \{ V_{g31}, V_{g32}, V_{g33} \}$ – змінення початкового етапу CI/CD процесу.

V_{g31} – змінення образу для контейнера в якому запускається CI/CD процес.

V_{g32} – змінення GIT конфігурації для підключення до GitLab репозиторію.

V_{g33} – змінення конфігурації ініціалізації Terraform для підключення до GitLab артефакторі.

$V_{g4} = \{ V_{g41}, V_{g42} \}$ – змінення етапу Terraform валідації для перевірки конфігураційних файлів Terraform на помилки.

V_{g41} – змінення виклику команди Terraform валідації.

V_{g42} – змінення правил запуску етапу Terraform валідації.

$V_{g5} = \{V_{g51}, V_{g52}, V_{g53}\}$ – блок змінення етапу планування Terraform скрипта для створення плану змін в інфраструктурі Terraform провайдерів.

V_{g51} – змінення виклику команди Terraform планування.

V_{g52} – змінення артефакту з планом, який створюється етапом Terraform планування.

V_{g53} – змінення правил запуску етапу Terraform планування.

$V_{g6} = \{V_{g61}, V_{g62}\}$ – змінення етапу розгортання Terraform інфраструктурних змін в інфраструктурі Terraform провайдерів.

V_{g61} – змінення виклику команди Terraform розгортання.

V_{g62} – змінення правил запуску етапу Terraform розгортання.

3. Скрипт Ansible [18]. $R3 = \{<IPa>, <SSH>, V_{a2}, V_{a3}, V_{a4}\}$.

IPa – зазначення груп віртуальних машин з їх приватними IP адресами.

SSH – створення приватного SSH ключа для SSH з'єднання з віртуальними машинами.

V_{a1} – шифрування приватного SSH ключа, використовуючи Ansible Vault.

V_{a2} – змінення групових змінних Ansible для використання змінних на окремих групах віртуальних машин.

$V_{a3} = \{V_{a31}, V_{a32}, V_{a33}, V_{a34}, V_{a35}, V_{a36}\}$ – змінення Ansible ролей для описання конфігурацій.

V_{a31} – змінення конфігураційних файлів docker ролі для встановлення docker на віртуальні машини.

V_{a32} – змінення конфігураційних файлів node_exporter ролі для запуску docker контейнеру із node exporter на віртуальні машини.

V_{a33} – змінення конфігураційних файлів cadvisor ролі для запуску docker контейнеру із cadvisor на віртуальні машини.

V_{a34} – змінення конфігураційних файлів prometheus ролі для запуску docker контейнеру із prometheus та Alertmanager на віртуальну машину.

V_{a35} – змінення конфігураційних файлів web ролі для запуску docker контейнеру із pgAdmin на віртуальну машину.

V_{a36} – змінення конфігураційних файлів db ролі для запуску docker контейнеру із postgresSQL DB на віртуальну машину.

$V_{a4} = \{V_{a41}, V_{a42}\}$ – змінення схеми розгортання Ansible конфігурації.

V_{a41} – змінення групи віртуальних машин на яких буде розгортатися Ansible конфігурація.

V_{a42} – змінення переліку Ansible ролей, які будуть запускатися для окремої групи машин.

4. Скрипт Terraform. $R4 = \{V_{t1}, V_{t2}, V_{t3}, V_{t4}, V_{t5}, V_{t6}, V_{t7}, V_{t8}\}$.

$V_{t1} = \{V_{t11}, V_{t12}, V_{t13}, V_{t14}\}$ – змінення Terraform провайдерів (AWS, PagerDuty) для підключення до AWS, PagerDuty серверів.

V_{t11} – змінення шляху у Terraform Registry до провайдерів.

V_{t12} – змінення версії провайдерів.

V_{t13} – змінення PagerDuty токена для підключення до PagerDuty серверу.

V_{t14} – змінення конфігурації AWS провайдера для використання ролі доступу (IAM) при підключенні до AWS серверу.

V_{t2} – оголошення Terraform змінних для використання їх в розгортанні інфраструктури.

$V_{t3} = \{V_{t31}, V_{t32}, V_{t33}\}$ – змінення bash скрипту для початкової ініціалізації Ansible віртуальної машини.

V_{t31} – встановлення Ansible сервісу.

V_{t32} – викачування Ansible конфігураційних файлів із GitHub репозиторію.

V_{t33} – дешифрування приватного SSH ключа для SSH з'єднання з віртуальними машинами.

V_{t4} – створення публічного SSH ключа для надання його віртуальним машин з якими буде з'єднуватися Ansible.

$V_{t5} = \{V_{t51}, V_{t52}, V_{t53}\}$ – створення мережевої частини проєкту в AWS.

V_{151} – створення AWS VPC (віртуальна приватна мережа) та публічних та приватних підмереж для розміщення віртуальних машин.

V_{152} – створення AWS SG (група безпеки) кожної віртуальної машини для зазначення дозволених портів на вхід/вихід.

V_{153} – створення NAT та Internet шлюзів для доступу у інтернет із приватної мережі та для доступу у інтернет із VPC відповідно.

$V_{16} = \{V_{161}, V_{162}\}$ – створення AWS IAM доступу для ansible віртуальної машини.

V_{161} – створення AWS IAM ролі з можливістю використовувати менеджер сесій для підключення до віртуальної машини.

V_{162} – створення AWS IAM профілю для віртуальної машини з роллю V_{161} .

$V_{17} = \{V_{171}, V_{172}, V_{173}, V_{174}, V_{175}\}$ – створення AWS EC2 (віртуальна машина). Для всіх EC2 зазначається тип машини, AMI (образ машини в Amazon), мережеві об'єкти V_{15} .

V_{171} – створення AWS web EC2 із публічним SSH ключем V_{14} .

V_{172} – створення AWS prom EC2 із публічним SSH ключем V_{14} .

V_{173} – створення AWS db EC2 із публічним SSH ключем V_{14} .

V_{174} – створення AWS ansible EC2 із файлом ініціалізації V_{13} та IAM профілем V_{16} .

$V_{18} = \{V_{181}, V_{182}, V_{183}, V_{184}, V_{185}\}$ – створення PagerDuty інфраструктури для надходження повідомлень про інциденти.

V_{181} – створення PagerDuty команди для реагування на повідомлення по ескалаціям.

V_{182} – створення PagerDuty політики ескалації із зазначенням користувачів на рівнях ескалації та команди V_{181} .

V_{183} – створення PagerDuty сервісу із зазначенням налаштувань повідомлень по сервісу та політики ескалації V_{182} .

V_{184} – створення PagerDuty інтеграції із зазначенням типу інтеграції та сервісу V_{183} .

5. Розгортання Terraform інфраструктури $R5 = \{V_{dt1}\}$.

V_{dt1} – запустити процес CI/CD у GitLab для розгортання Terraform інфраструктури.

6. Розгортання Ansible конфігурації $R6 = \{V_{da1}, V_{da2}\}$.

V_{da1} – під'єднання до Ansible EC2, користуючись системним менеджером.

V_{da1} – запустити Ansible скрипт для розгортання Ansible конфігурації.

2.6 Метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Засновуючись на загальних вимогах автоматизації моніторингу інфраструктур можливо побудувати загальний опис методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.

Загальний опис методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем представлений у вигляді послідовності: $R1 \rightarrow R2 \rightarrow R3 \rightarrow R4 \rightarrow R5 \rightarrow R6$.

Крок 1. Створення та перевірка аккаунтів, налаштування віртуальних машин $R1 = \{<AWS \text{ аккаунт}>, <GitLab \text{ сервер}>, <PagerDuty \text{ аккаунт}>\}$ (таблиця 2.4).

Крок 2. Оголошення змінних оточення CI/CD процесу для використання при конфігурації CI/CD та надання до змінних оточення Terraform (Vt). визначення порядку виконання етапів CI/CD, змінення образу для контейнера в якому запускається CI/CD процес. Змінення GIT конфігурації для підключення до GitLab репозиторію, змінення конфігурації ініціалізації Terraform для підключення до GitLab артефактові, формування правил запуску етапу Terraform валідації та етапу Terraform планування, формування правил

запуску етапу Terraform розгортання. $R2 = \{V_{g1}, V_{g2}, V_{g3}, V_{g4}, V_{g5}, V_{g6}\}$. Виконується за допомогою скрипту gitlab-ci (таблиця 2.5).

Крок 3. Формування простору та конфігурації IP адрес, налаштування SSH (створення SSH ключа, шифрування приватного SSH ключа), формування Ansible ролей для конфігурування docker, node_exporter, cadvisor, Prometheus, web, db. Формування ролей для запуску контейнерів з pgAdmin, postgresQL DB. Змінення схеми розгортання Ansible ролей та конфігурації для віртуальних машин, $R3 = \{<IPa>, <SSH>, V_{a2}, V_{a3}, V_{a4}\}$. Виконується за допомогою скрипту Ansible (таблиця 2.6).

Крок 4. Налаштування Terraform провайдерів, Створення сценаріїв використання Ansible сервісів, створення мережевої частини проєкту в AWS (AWS VPC, AWS SG, AWS публічних та приватних під мереж, NAT шлюзу, Internet шлюзу), формування ролей для підключення до віртуальної машини, налаштування віртуальних машин у AWS, налаштування процедур для контролю повідомлень про інциденти (PagerDuty). $R4 = \{V_{t1}, V_{t2}, V_{t3}, V_{t4}, V_{t5}, V_{t6}, V_{t7}, V_{t8}\}$ Виконується за допомогою скрипту Terraform (таблиця 2.7).

Крок 5. Розгортання Terraform інфраструктури $R5 = \{V_{dt1}\}$ (рисунок 2.20).

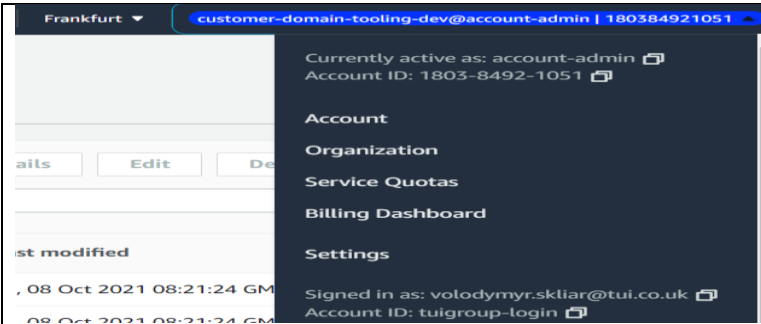
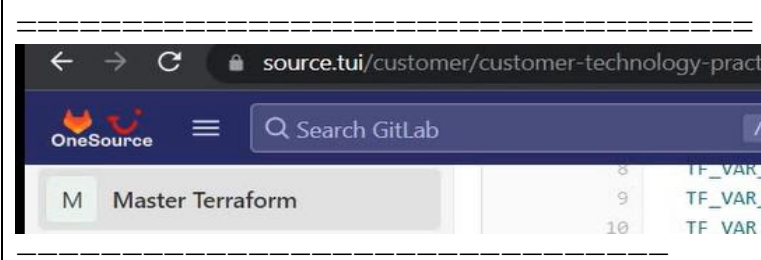
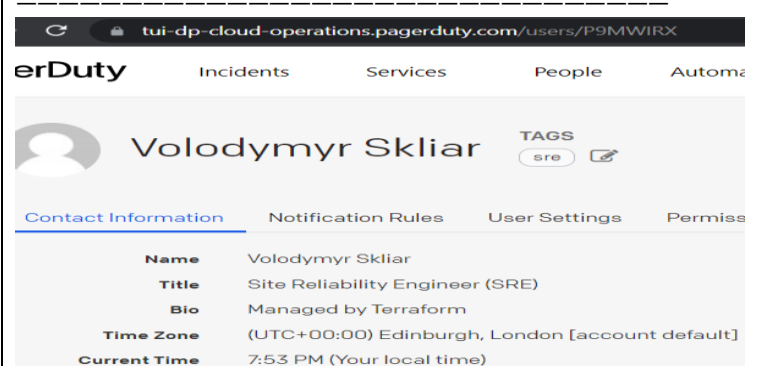
Крок 6. Розгортання Ansible конфігурації. $R6 = \{V_{da1}, V_{da2}\}$ (рисунки 2.21 – 2.22).

Після виконання шести кроків методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем створюється CI/CD процес для розгортання Terraform інфраструктури. Створюється інфраструктура для роботи з Ansible. Формується бібліотека скриптів (gitlab-ci, Terraform, Ansible). Створюється хмарна інфраструктура для розміщення всіх компонентів системи. Розгортається web-додаток та БД до нього. Розгортається моніторинг системи, котрий можливо масштабувати.

2.7 Пояснення до методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем (передумови, формування бібліотеки скриптів)

На таблицях 2.4 – 2.7 представлені пояснення до кроків методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем. А саме, передумова (Крок 1) та формування бібліотеки скриптів (Кроки 2, 3, 4).

Таблиця 2.4 – Метод Крок 1, Передумови

	<p>Авторизований AWS аккаунт</p> <p>Зазначається: назва та ID аккаунту, ім'я користувача, AWS регіон та роль користувача.</p>
	<p>Підготовлений GitLab сервер із налаштованими агентами для запуску CI/CD процесу.</p>
	<p>Авторизований PagerDuty аккаунт.</p> <p>Зазначається: ім'я, посада, теги, часовий пояс та додаткова інформація користувача</p>

Таблиця 2.5 – Метод Крок 2, GitLab скрипт для автоматизації розгортання інфраструктури

<pre> variables: TERRAFORM_DIRECTORY_NAME: "infra" TERRAFORM_VERSION: 1.0.5 TF_VAR_AWS_DEPLOYMENT_ROLE: \${AWS_ROLE_NAME} TF_VAR_AWS_REGION: \${AWS_REGION} TF_VAR_AWS_ACCOUNT_ID: \${CUSTOMER_TOOLS_DEV_ACCOUNT_ID} TF_VAR_AWS_ACCOUNT_NAME: \${CUSTOMER_TOOLS_DEV_ACCOUNT_NAME} TF_VAR_AWS_EXTERNAL_ID: \${CUSTOMER_TOOLS_DEV_EXTERNAL_ID} TF_VAR_pagerduty_token: \${PAGERDUTY_TOKEN} TF_VAR_PGADMIN_ADMIN_PASSWORD: \${PGADMIN_ADMIN_PASSWORD} TF_VAR_POSTGRES_ADMIN_PASSWORD: \${POSTGRES_ADMIN_PASSWORD} TF_VAR_ANSIBLE_VAULT_KEY: \${ANSIBLE_VAULT_KEY} ===== stages: - tf-test - tf-plan - tf-deploy ===== .tf_base: image: name: hashicorp/terraform:\${TERRAFORM_VERSION} entrypoint: [""] before_script: - gitconfig --global url."https://gitlab-ci- token:\${CI_JOB_TOKEN}@source.tui".insteadOf https://source.tui - gitconfig --global url."https://gitlab-ci- token:\${CI_JOB_TOKEN}@source.tui".insteadOf ssh://git@ssh.source.tui - export TF_HTTP_USERNAME="gitlab-ci-token" - export TF_HTTP_PASSWORD="\${CI_JOB_TOKEN}" - export TERRAFORM_ADDRESS=\${CI_API_V4_URL}/projects/\${CI_PROJECT_ID}/terraform/state/ \${CI_PROJECT_NAME}-prod - cd \${TERRAFORM_DIRECTORY_NAME}/ - terraform init --backend-config="address=\${TERRAFORM_ADDRESS}" --backend-config="lock_address=\${TERRAFORM_ADDRESS}/lock" --backend-config="unlock_address=\${TERRAFORM_ADDRESS}/lock" --backend-config="lock_method=POST" --backend-config="unlock_method=DELETE" --backend-config="retry_wait_min=5" ===== </pre>	<p>Блок оголошення змінних, котрі будуть використовуватися при розгортанні Terraform. TF_VAR префікс дозволяє використовувати цю змінну, як змінну оточення у Terraform.</p> <p>Блок оголошення етапів розгортання</p> <p>Блок оголошення розширення, котре потім буде використовувати в усіх етапах розгортання. В цьому розширенні описано образ докеру з Terraform, налаштування GIT конфігурації та ініціалізація Terraform з методом зберігання tfstate файлу в GitLab артефактах.</p>	<pre> tf-validate: extends: .tf_base stage: tf-test script: - terraform validate rules: - if: \$CI_COMMIT_BRANCH == "main" changes: - infra/* - infra/scripts/* ===== tf-plan: extends: .tf_base stage: tf-plan tags: - customer - docker script: - terraform plan -out plan.tfplan artifacts: paths: - \${TERRAFORM_DIRECTORY_NAME}/plan.tfplan expire_in: 1 week rules: - if: \$CI_COMMIT_BRANCH == "main" changes: - infra/* - infra/scripts/* ===== tf-apply: extends: .tf_base tags: - customer - docker stage: tf-deploy script: - terraform apply -input=false -auto-approve ./plan.tfplan rules: - if: \$CI_COMMIT_BRANCH == "main" changes: - infra/* - infra/scripts/* when: manual </pre>	<p>Блок виконання етапу валідації Terraform конфігурації – це є тестуванням.</p> <p>Блок виконання етапу планування змін в конфігурації та зберігаємо план в артефакторі.</p> <p>Блок виконання етапу розгортання змін. Переглянувши план, приймаємо рішення розгорнути зміни чи ні. Зміни розгортаються спираюсь на побудований у минулому блоці план.</p>
--	---	--	---

Таблиця 2.6 – Метод Крок 3, Ansible скрипт для розгортання конфігурацій на віртуальні машини

<pre> - hosts: all become: true gather_facts: no roles: - { role: docker_role } - { role: node_exporter_role } - { role: cadvisor_role } - hosts: prometheus become: true gather_facts: no roles: - { role: prometheus_role } - hosts: pgadmin become: true gather_facts: no roles: - { role: web_role } - hosts: postgres become: true gather_facts: no roles: - { role: db_role } ===== [prometheus] 10.140.26.80 [pgadmin] 10.140.26.70 [postgres] 10.140.26.10 ===== --- ansible_user : ec2-user postgres_admin_password : db_password ... ===== [defaults] host_key_checking = false inventory = ./hosts.txt private_key_file = ./ansible </pre>	<p>Блок з описанням виклику ролей для наявних груп віртуальних машин. На всіх машинах команди виконуються від root користувача (become: true). З усіх машин не збирається додаткова інформація про конфігурацію цих машин (gather_facts: no). На всіх машинах виконуються docker, node exporter та , cadvisor полі. На кожній окремій машині виконується специфічна роль: web, prometheus, db.</p> <p>Блок з даними про статичні приватні ір адреси віртуальних машин, на яких будуть виконуватися конфігураційні зміни.</p> <p>Блок з груповими змінними для postgres групи. Пароль адміна бази даних вписується під час ініціалізації ansibleмашини інструментом Terraform.</p> <p>Блок з конфігурацією Ansible. Визначається не перевіряти ключ при доступі до залежних машин (host_key_checking), шлях до файлу із приватними ір адресами (inventory)та шлях до приватного ключа ssh (private_key_file).</p>	<pre> \$ANSIBLE_VAULT;1.1;AES256 31343333336336664376163353031 6166386633353336366131636330 3730633965316163393862356461 3764356231346663646233326339 35363865326636660a3865373132 636365656334383464 ===== - name: Install the latest version of Docker yum: name: docker state: latest - name: Start docker systemd: state: started name: docker enabled: yes - name: Install docker python package pip: name: docker version: 2.0.2 ===== - name: Copy prometheus config file template: src: prometheus.yml.j2 dest: prometheus.yml ===== - name: Copy file with prom container rules ansible.builtin.copy: src: container.rules dest: container.rules ===== - name: Run an prometheus container docker_container: name: prometheus image: prom/prometheus ports: - "80:9090" volumes: - "/prometheus.yml:/etc/prometheus/prometheus.yml" - "/container.rules:/etc/prometheus/container.rules" - "/host.rules:/etc/prometheus/host.rules" </pre>	<p>Блок із частиною засекреченого приватного ssh ключа, за допомогою якого ansible підключається до інших машин. Ключ засекречено за допомогою ansible vault. Ключ розсекрчується під час ініціалізації ansibleмашини інструментом Terraform.</p> <p>Блок з викачуванням docker останньої версії, запуском dockerta викачуванням ірпакету із docker версії 2.0.2.</p> <p>Блок з копіюванням шаблону із конфігурацією prometheus із локального сховища ansibleна віддалену машину. Під час копіюванн у шаблон записуються необхідна інформація із групових змінних prometheus.</p> <p>Блок з копіюванням файлу правил для моніторингу контейнерів із локального сховища ansibleна віддалену машину.</p> <p>Блок запуску docker контейнеру із prometheus. Також азначається порт, на якому буде працювати prometheus (ports),та томи (volumes) із шляхом до конфігураційними файлами prometheus. Шлях до конфігураційних файлів зазначається у форматі «шлях до файлу на віртуальній машині prometheus»:«шлях до файлу який буде створено у контейнері ізprometheus ». Таким чином при зміні файлів на віртуальній машині і подальшим перезапуском контейнера конфігурація буде змінюватися.</p>
--	--	--	--

Таблиця 2.7 – Метод крок 4, Terraform скрипт для розгортання інфраструктури

<pre> variable "AWS_ACCOUNT_ID" { description = "AWS Account ID (number)" type = string } ===== variable "availability_zones" { type = list(string) default = ["eu-central-1a", "eu-central-1b"] description = "List of availability zones" } ===== terraform { required_version = ">=0.14" backend "http" {} required_providers { aws = { source = "hashicorp/aws" version = "~> 4.0" } } pagerduty = { source = "PagerDuty/pagerduty" version = "2.6.1" } } ===== provider "aws" { region = var.AWS_REGION assume_role { role_arn = var.AWS_DEPLOYMENT_ROLE != null ? "arn:aws:iam::\${var.AWS_ACCOUNT_ID}:role/\${var.AWS_DEPLOYMENT_ROLE}" : "" external_id = var.AWS_EXTERNAL_ID session_name = "terraform@\${var.AWS_ACCOUNT_NAME}" } } ===== provider "pagerduty" { token = var.pagerduty_token } ===== </pre>	<p>Блок оголошення змінної, значення якої передається з змінних оточення.</p> <p>Блок оголошення змінної, значення якої зазначене по замовчуванню.</p> <p>Блок оголошення Terraform конфігурації із зазначенням необхідної версії та необхідних версій провайдерів. Також у цьому блоці визначається тип сховища файлу стану Terraform.</p> <p>Блок оголошення AWS провайдеру, у якому зазначається регіон та IAM роль для агента, що розгортає Terraform.</p> <p>Блок оголошення PagerDuty провайдеру із зазначенням токена для доступу у PagerDuty.</p>	<pre> resource "pagerduty_team" "main" { name = "SRE-team" description = "Team for SRE notifications" } resource "pagerduty_escalation_policy" "main" { name = "SRE Escalation Policy" num_loops = 2 teams = [pagerduty_team.main.id] rule { escalation_delay_in_minutes = 10 target { type = "user_reference" id = data.pagerduty_user.main.id } } } ===== resource "aws_key_pair" "ansible" { key_name = "ansible-key" public_key = file("\${path.module}/ansible.pub") } module "web_ec2" { source = "terraform-aws-modules/ec2-instance/aws" version = "~> 4.1" name = "\${var.environment}-pgadmin" ami = "ami-0a1ee2fb28fe05df3" instance_type = "t2.micro" monitoring = true vpc_security_group_ids = [module.web_server_sg.security_group_id, module.system_sg.security_group_id] subnet_id = module.vpc.public_subnets[0] private_ip = var.web_private_ip key_name = aws_key_pair.ansible.key_name tags = { Terraform = "true" Environment = var.environment } } </pre>	<p>Блок створення частини PagerDuty інфраструктури, а саме команди та політики ескалації для доставки сповіщення про інцидент.</p> <p>Блок створення частини AWS інфраструктури. Створюється ssh ключ для доступу із Ansible машини до інших машин по ssh протоколу. Також створюється віртуальна машина для web серверу, де буде знаходитися pgadmin. Для створення віртуальної машини та залежностей для її роботи використовується офіційний модуль Terraform. У цей модуль передаються усі необхідні змінні.</p>
--	---	---	--

2.8 Пояснення до методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем (розгортання змін)

Остання частина методу це розгортання Terraform інфраструктурних та Ansible конфігураційних змін. Для розгортання Terraform інфраструктурних змін використовується GitLab CI (рисунок 2.20).

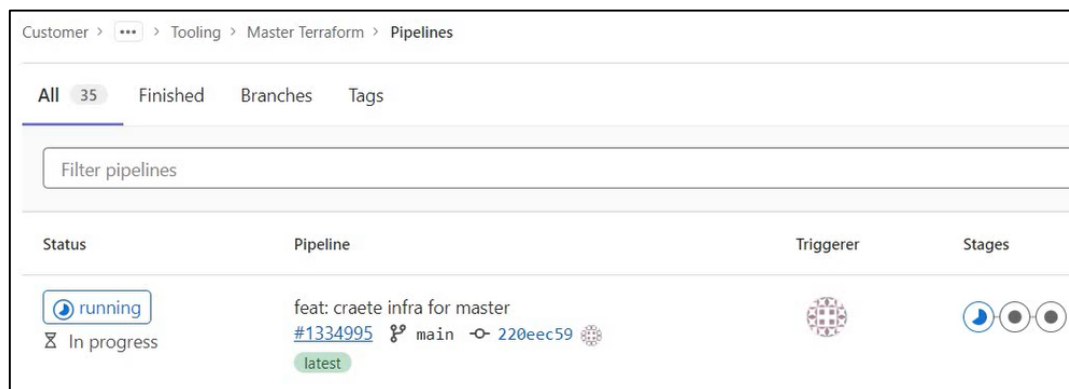


Рисунок 2.20 – Метод Крок 5, запуск процесу CI/CD у GitLab для розгортання Terraform інфраструктури

На рисунку 2.20 зазначається:

- статус розгортання із затracеним часом на розгортання і його результатом;
- гілка з якої запускається розгортання;
- користувач, який запустив розгортання;
- етапи розгортання і їх результати.

Для розгортання Ansible конфігураційних змін потрібно під'єднатися до Ansible віртуальної машини (рисунок 2.21).

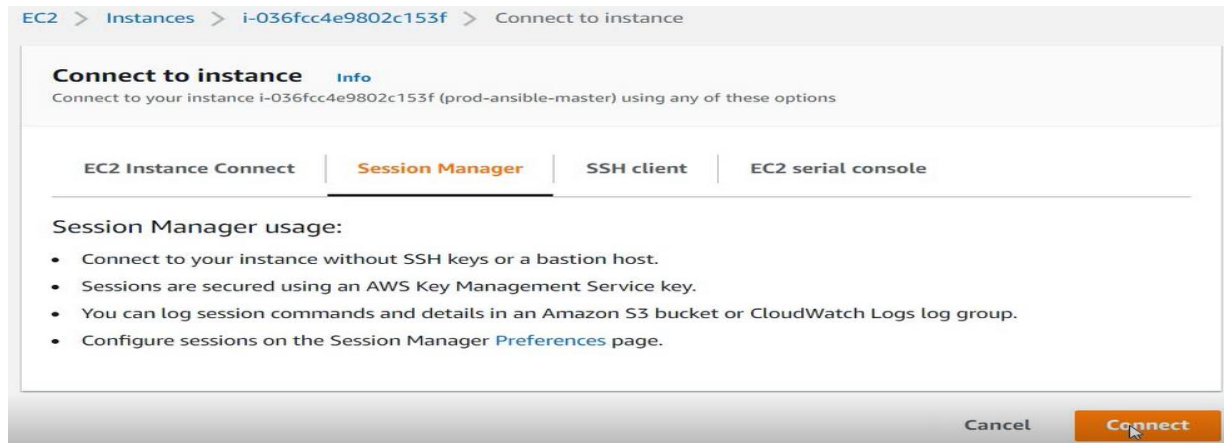


Рисунок 2.21 – Метод Крок 6, підключення до віртуальної машини з Ansible

Для підключення до віртуальної машини з Ansible на рисунку 2.21 використовується AWS менеджер сесій (Session Manager). За допомогою цього менеджера підключення до віртуальної машини відбувається без відкритого 22 порту та SSH протоколу, що покращує безпеку віртуальної машини та не потребує додаткових конфігурувань мережевої частини.

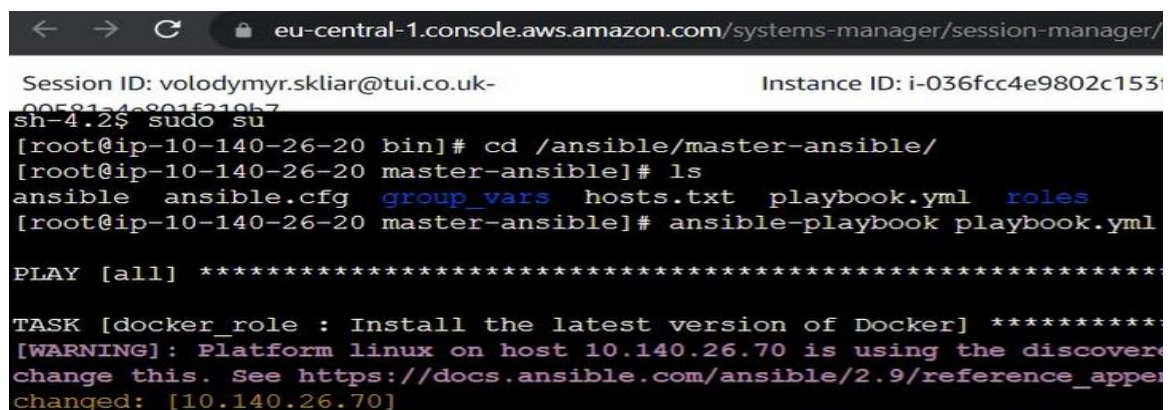


Рисунок 2.22 – Метод Крок 6, запуск Ansible скрипта для розгортання Ansible конфігурації

На рисунку 2.22 Ansible директорія складається з:

- ansible – файл з приватним SSH ключем для під'єднання до залежних віртуальних машин;

- `ansible.cfg` – файл з конфігурацією Ansible для зазначення системних змінних;
- `hosts.txt` – файл з групами віртуальних машин із їх приватними IP адресами для створення ієрархії;
- `group_vars` – папка з файлами з груповими змінними для динамічного менеджменту та передачі зашифрованих змінних;
- `roles` – папка з файлами конфігурації Ansible ролей для створення конфігурацій;
- `playbook.yml` – файл розгортання Ansible конфігурацій.

Запуск Ansible файлу розгортання конфігурацій відбувається за допомогою команди `ansible-playbook`.

2.9 Terraform інфраструктура методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Для розуміння процесу створення Terraform інфраструктури створені діаграми послідовності.

Діаграми послідовності внесення змін до скриптів конфігурації Terraform інфраструктури зображена на рисунку 2.23.

На рисунку 2.23 користувач змінює Terraform конфігурацію у GitLab web-клієнті. Після цього запускається 1 етап CI/CD процесу – `validation`. Після валідації конфігураційних файлів Terraform на GitLab WebServer користувач отримує результат валідації.

Після валідації запускається етап планування, виконується Terraform Init команда (рисунк 2.24), отримується статус інфраструктури від Terraform провайдерів (PagerDuty, AWS), створюється план змін та зберігається до артефакторі. Користувач отримує Terraform план.

Після планування запускається етап розгортання, виконується Terraform Init команда (рисунок 2.24), блокується файл зі станом Terraform інфраструктури для унеможливлення паралельних розгортань, отримується Terraform план з артефакторі та розгортаються зміни на Terraform провайдерах. Після цього розблоковується та оновлюється файл стану Terraform, користувач отримує результат розгортання.

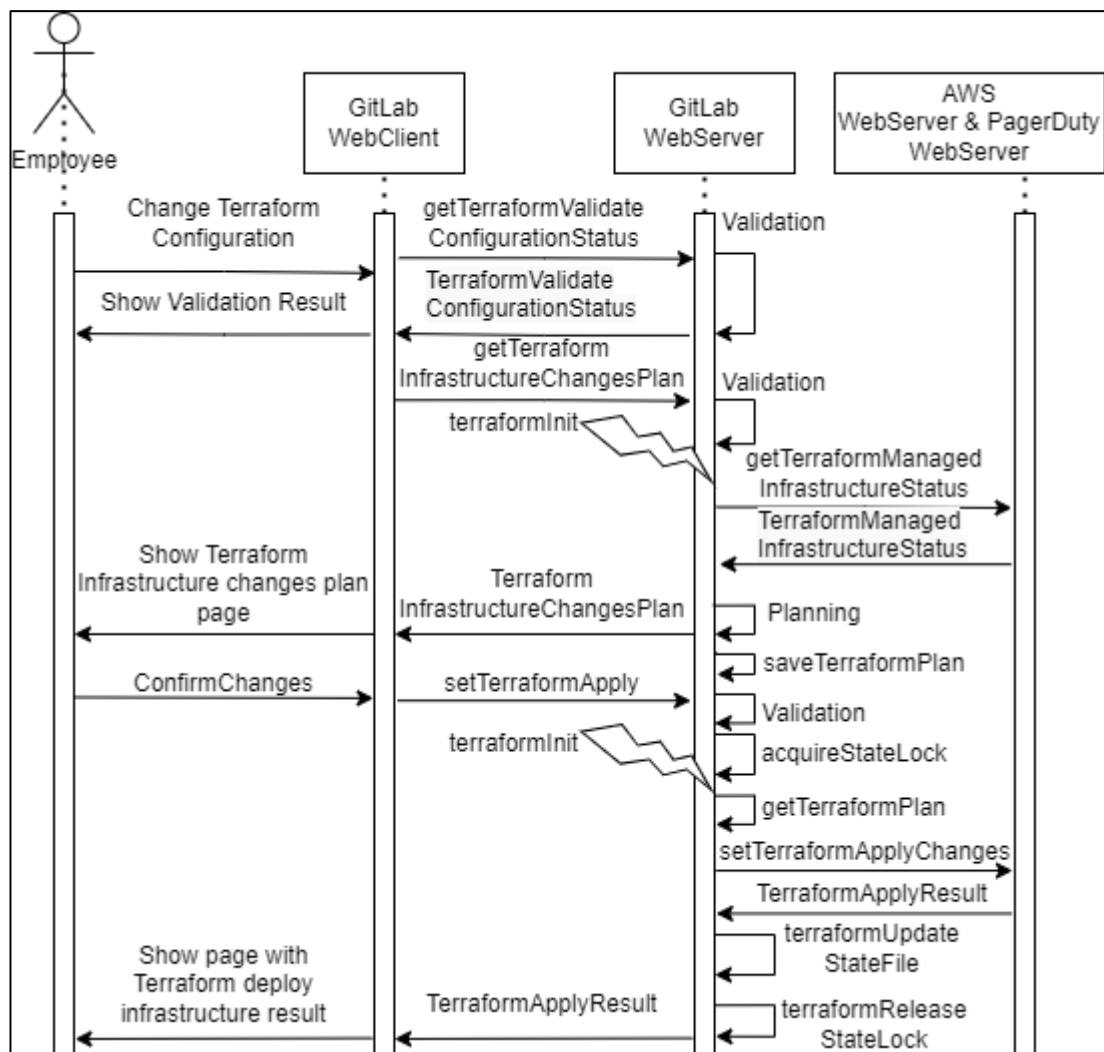


Рисунок 2.23 – Діаграма послідовності внесення змін до скриптів конфігурації Terraform інфраструктури

На рисунку 2.24 зображена діаграма послідовності команди Terraform Init.

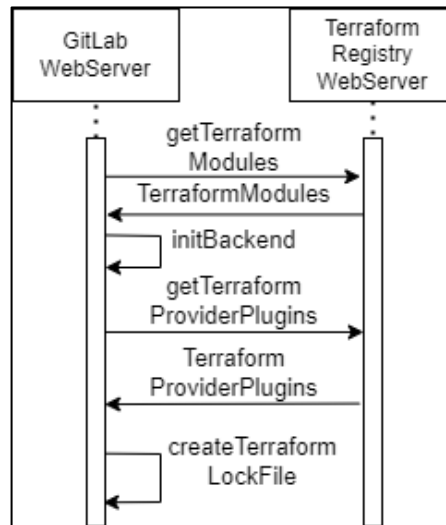


Рисунок 2.24 – Діаграма послідовності команди Terraform Init

На рисунку 2.24 розглядається команда Terraform Init, яка виконується задля підготовки робочої директорії до роботи з Terraform. Під час виконання команди викачуються Terraform модулі та плагіни із Terraform Registry, ініціалізується місце сховища файлу із станом Terraform інфраструктури. Створюється Terraform файл (lock file) із хешами та версіями завантажених плагінів для затвердження версій для Terraform Init команди. Lock file не зберігається після проходження CI/CD процесу, оскільки версії провайдерів та модулів зазначені у Terraform інфраструктурі.

Інфраструктура, що створюється інструментом Terraform згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем зображена на рисунку 2.25. Інфраструктура на рисунку 2.25 поділяється на підготовлену (GitLab CI, GitHub) та створену інструментом Terraform (AWS, PagerDuty інфраструктура).

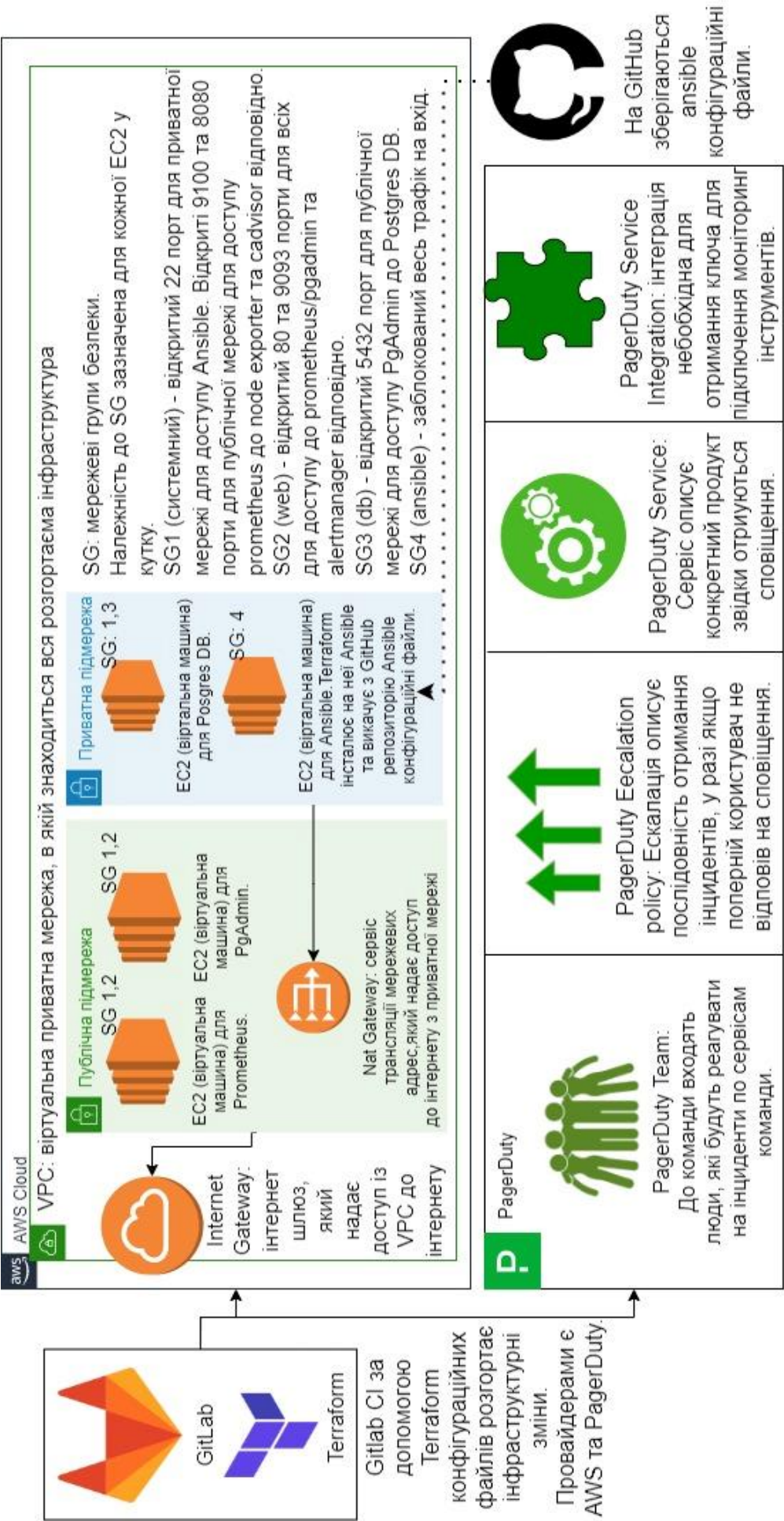


Рисунок 2.25 – Terraform інфраструктура згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

2.10 Ansible інфраструктура методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Схема конфігурації, що створюється інструментом Ansible згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем зображена на рисунку 2.26.

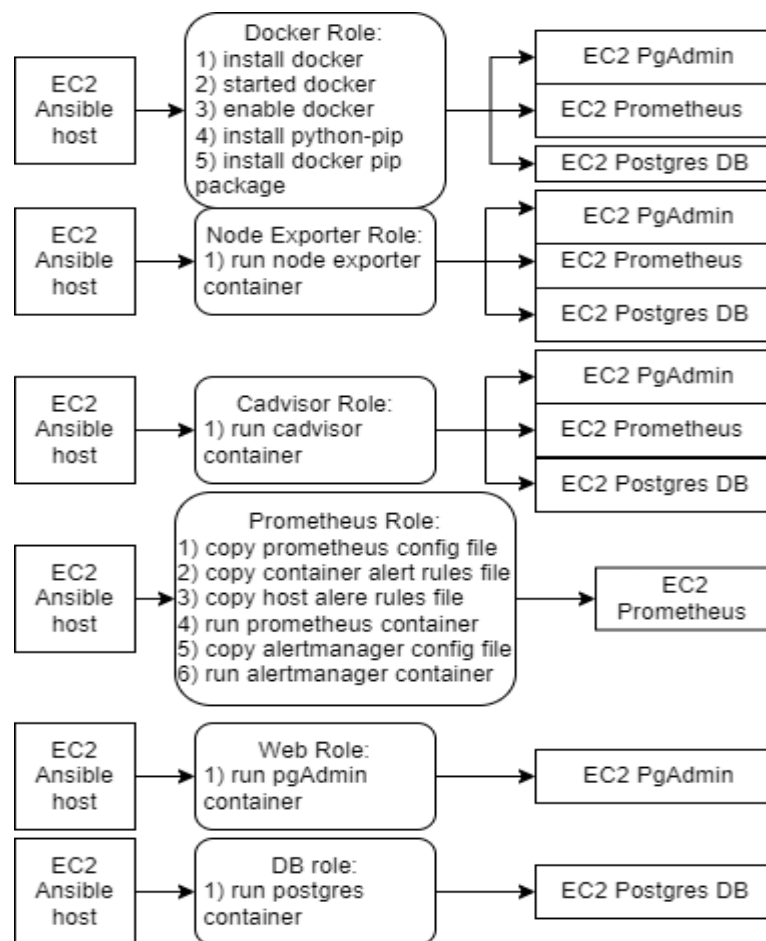


Рисунок 2.26 – Ansible конфігурація згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Схема Ansible конфігурації на рисунку 2.26 складається з ролей, а саме:

- `docker_role` – встановлює `docker` та `python docker`;

- node_exporter_role – запускає контейнер з node експортером;
- cadvisor_role – запускає контейнер з cadvisor експортером;
- prometheus_role – запускає prometheus та alertmanager контейнери та завантажує необхідні конфігураційні файли для них;
- web_role – запускає контейнер з pgadmin, який слугує web сервером;
- db_role – запускає контейнер с postgresSQL базою даних.

Діаграма послідовності розгортання Ansible конфігурації на залежні віртуальні машини зображена на рисунку 2.27.

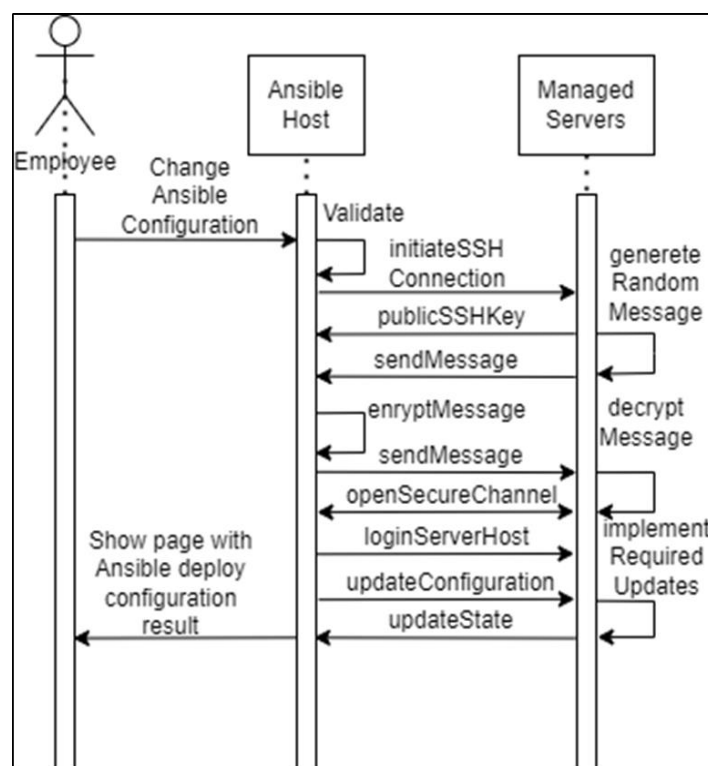
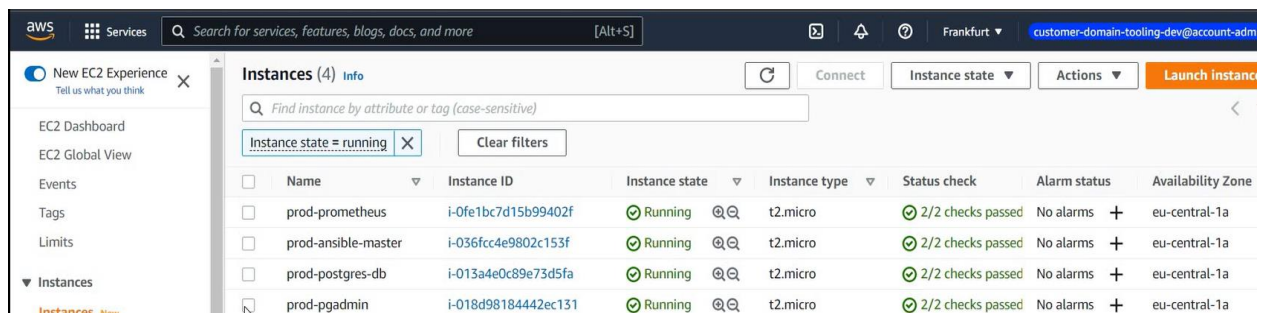


Рисунок 2.27 – Діаграма послідовності розгортання Ansible конфігурації

На рисунку 2.27 користувач змінює Ansible конфігурацію, після чого Ansible на Ansible VM валідує зміни та створює SSH з'єднання із залежними серверами. Після цього на залежні сервери завантажується необхідна конфігурація. Після встановлення необхідних оновлень процес закінчується та користувач отримує результат зміни конфігурації.

2.11 Технології реалізації інфраструктури методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Розгорнута AWS інфраструктура [19] із переліком віртуальних машин (EC2) та їх характеристиками зображена на рисунку 2.28.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
prod-prometheus	i-0fe1bc7d15b99402f	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a
prod-ansible-master	i-036fcc4e9802c153f	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a
prod-postgres-db	i-013a4e0c89e73d5fa	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a
prod-pgadmin	i-018d98184442ec131	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a

Рисунок 2.28 – Розгорнута AWS інфраструктура

Префікс prod в назві EC2 позначає те, що машина створена на головному оточенні. У кожній EC2 є власний ID. EC2 може знаходитися у 7 станах. Pending – коли EC2 запускається. Running – коли EC2 працює. Rebooting – коли EC2 перезавантажується. Stopping – коли EC2 зупиняється або впадає в гібернацію. Stopped – коли EC2 зупинена. Shutting-down – коли EC2 готується до видалення. Terminated – коли EC2 видалена. На рисунку 2.28 всі EC2 працюють.

Перелік створених віртуальних машин на рисунку 2.28:

- prod-prometheus – EC2 з контейнерами Prometheus та Alertmanager;
- prod-ansible-master – EC2 з встановленим ansible, з якого запускається розгортання ansible конфігурації;
- prod-postgres-db – EC2 з контейнером PostgreSQL базою даних;
- prod-pgadmin – EC2 з контейнером pgAdmin.

Значення характеристик створених EC2 зазначені у таблиці 2.8.

Таблиця 2.8 – характеристики EC2

Назва характеристики EC2	Значення характеристики EC2
Родина віртуальних машин, версія, розмір	t2.micro
Кількість віртуальної CPU	1
Кількість МеМ	1 GiB
Операційна система	Amazon Linux 2
Ємність сховища	8 GiB
Тип сховища	SSD
Регіон розгортання	центральна Європа, Франкфурт (eu-central-1)
Зона доступності	A (eu-central-1a)

Процес CI/CD розгортання Terraform інфраструктури у GitLab зображений на рисунку 2.29.

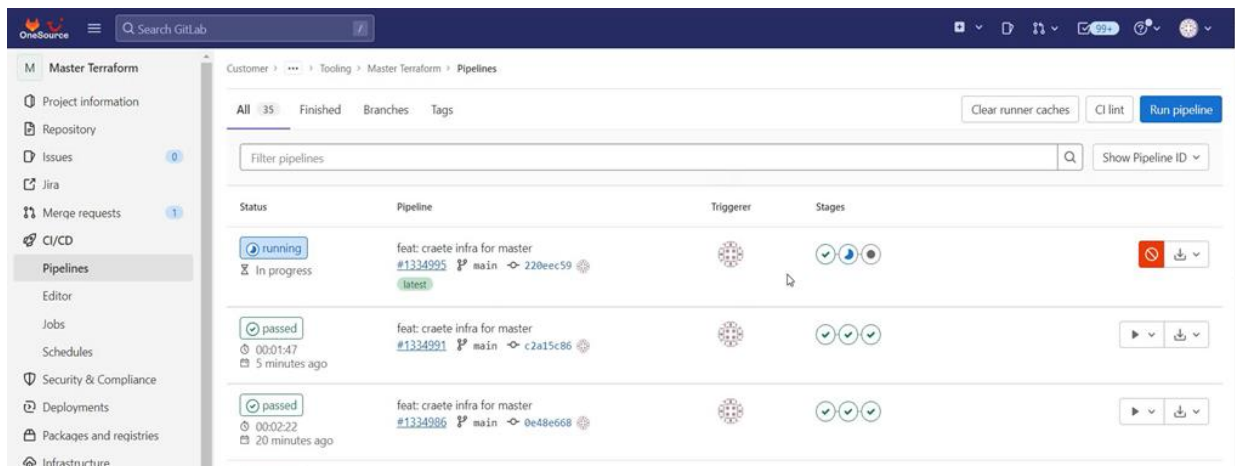


Рисунок 2.29 – Процес CI/CD розгортання Terraform інфраструктури у GitLab

Основними елементами GitLab, що використовуються, є Repository, Merge Requests, CI/CD, Infrastructure.

Repository – використано для збереження файлів з можливістю відслідкувати зміни та відмінити зміни.

Merge Requests – використано для додаткових перевірок якості коду та вимагання отримати дозвіл на розгортання змін.

CI/CD – використано для побудови схеми розгортання (pipeline).

Infrastructure – використано для зберігання файлу стану Terraform інфраструктури у GitLab.

Процес CI/CD розгортання складається із 3 етапів:

- валідації файлів Terraform (рисунок 2.30);
- планування змін в Terraform інфраструктурі (рисунок 2.31);
- розгортання Terraform змін (рисунок 2.32).

```

125 Terraform has created a lock file .terraform.lock.hcl to record the provider
126 selections it made above. Include this file in your version control repository
127 so that Terraform can guarantee to make the same selections by default when
128 you run "terraform init" in the future.
129 Terraform has been successfully initialized!
130 You may now begin working with Terraform. Try running "terraform plan" to see
131 any changes that are required for your infrastructure. All Terraform commands
132 should now work.
133 If you ever set or change modules or backend configuration for Terraform,
134 rerun this command to reinitialize your working directory. If you forget, other
135 commands will detect it and remind you to do so if necessary.
136 $ terraform validate
137 Success! The configuration is valid.

```

Рисунок 2.30 – Валідація файлів Terraform

Terraform validate – команда перевіряє конфігураційні файли в каталозі, посилаючись лише на конфігурацію та не звертаючись до будь-яких віддалених служб, таких як віддалений стан, API провайдера тощо.

Validate запускає перевірки, які перевіряють, чи конфігурація є синтаксично правильною та внутрішньо узгодженою, незалежно від будь-яких наданих змінних або наявного стану. Таким чином, це в першу чергу корисно для загальної перевірки багаторазових модулів, включаючи правильність назв атрибутів і типів значень.

Валідація пройшла успішно, Terraform конфігурація валідна.

Після планування змін, згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, Terraform планує створити 37 ресурсів та зберігає план до артефакторі (рисунок 2.30).

```

1132   + cidr_blocks           = [
1133     + "0.0.0.0/0",
1134   ]
1135   + description           = "alertmanager"
1136   + from_port             = 9093
1137   + id                   = (known after apply)
1138   + prefix_list_ids       = []
1139   + protocol              = "tcp"
1140   + security_group_id     = (known after apply)
1141   + self                  = false
1142   + source_security_group_id = (known after apply)
1143   + to_port               = 9093
1144   + type                  = "ingress"
1145   }
1146 Plan: 37 to add, 0 to change, 0 to destroy.
1147
1148 Saved the plan to: plan.tfplan
1149 To perform exactly these actions, run the following command to apply:
1150     terraform apply "plan.tfplan"

```

Рисунок 2.30 – Планування змін в Terraform інфраструктурі

Процес розгортання Terraform змін, згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, успішно створив 37 запланованих ресурсів (рисунок 2.31).

```


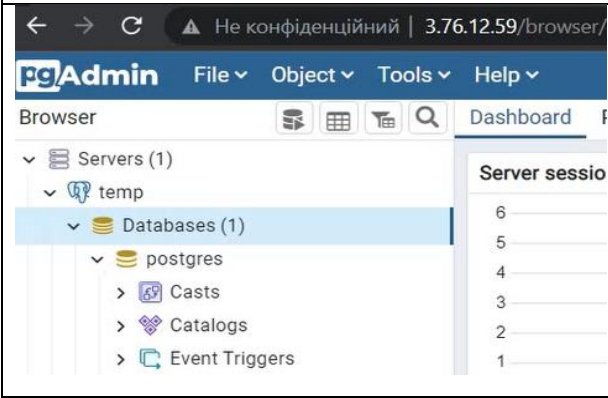
177 module.vpc.aws_route.private_nat_gateway[0]: Creation complete after 1s [id=r-rtb-0cf56fbce1f27e7da1080289494]
178 2022/10/16 15:30:39 [DEBUG] POST https://source.tui/api/v4/projects/20822/terraform/state/master-terraform-prod?ID=b3a
    Sebad-7f54-1590-8090-ae369764a634
179 2022/10/16 15:30:40 [DEBUG] DELETE https://source.tui/api/v4/projects/20822/terraform/state/master-terraform-prod/lock
180 Apply complete! Resources: 37 added, 0 changed, 0 destroyed.
182 Cleaning up project directory and file based variables
184 Job succeeded

```

Рисунок 2.31 – Розгортання Terraform змін

Створена PgAdmin та PostgreSQL БД інфраструктура, створена згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, зображена на таблиці 2.9.

Таблиця 2.9 – PgAdmin та PostgreSQL БД інфраструктура

	Логін сторінка PgAdmin інструменту, який моделює роботу web-серверу, згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
	PgAdmin web-сервер успішно під'єднався до віртуальної машини з PostgreSQL базою даних. Це означає, що сервіси розгорнулися правильно та мережевий зв'язок між ними присутній.

2.12 Технології реалізації моніторингу методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем

Prometheus інструмент, котрий розгорнутий згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, зображений на рисунках 2.32 – 2.33. На рисунку 2.32 зображені метрики від node експортеру про відсоток зайнятої оперативної пам'яті на віртуальних машинах у вигляді графіку.

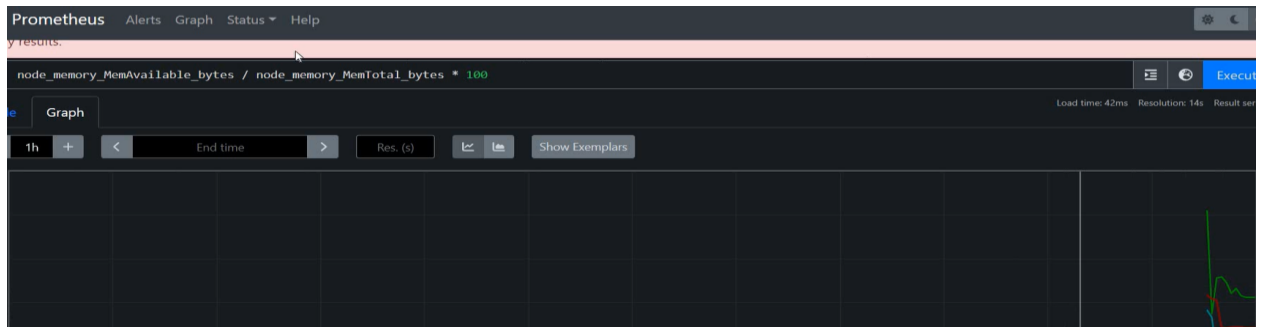


Рисунок 2.32 – Графік Prometheus метрик від node експортеру

На рисунку 2.33 зображені правила створення повідомлень, а саме:

- ContainerKilled, контейнер вимкнувся через переповнення оперативної пам'яті;
- ContainerCpuUsage, контейнер споживає багато ЦПУ;
- ContainerMemoryUsage, контейнер споживає багато оперативної пам'яті;
- HostHighCpuLoad, віртуальна машина споживає багато ЦПУ;
- HostOutOfMemory, віртуальна машина споживає багато оперативної пам'яті.

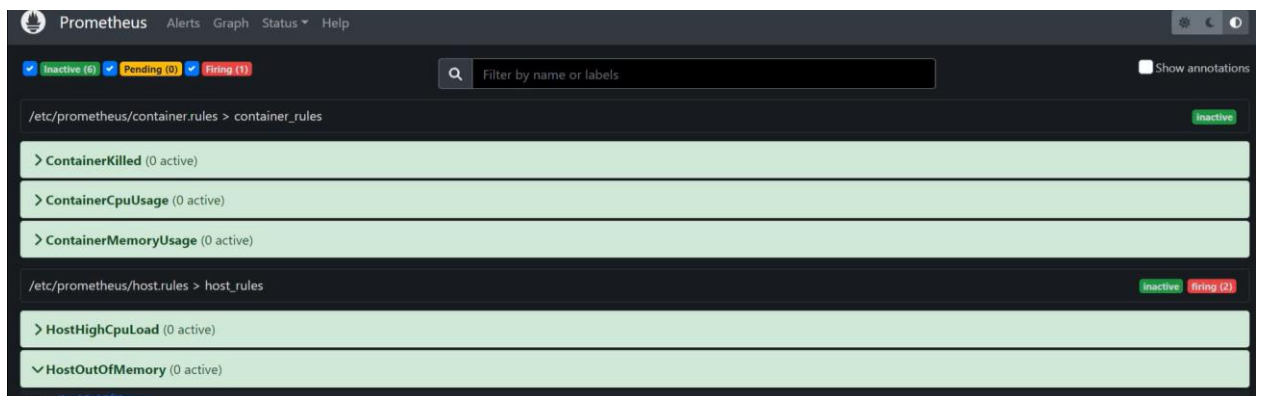


Рисунок 2.33 – Правила створення критичних повідомлень у Prometheus

Alertmanager інструмент, котрий розгорнутий згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, зображений на рисунку 2.34.

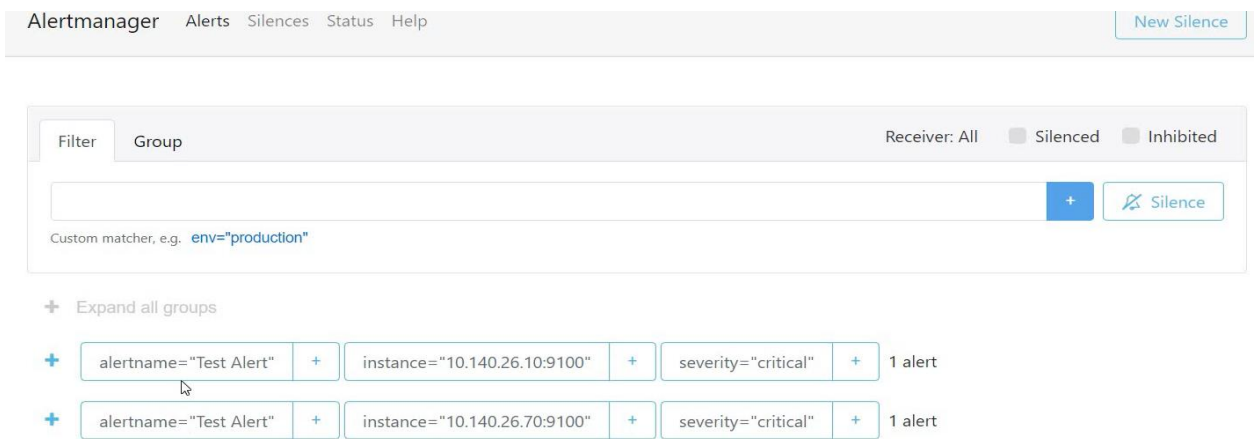


Рисунок 2.34 – Розгорнутий Alertmanager інструмент

До Alertmanager надійшли тестові повідомлення про інцидент. Повідомлення мають атрибути:

- ім'я критичного повідомлення;
- приватне IP віртуальної машини, з якого надійшло повідомлення;
- серйозність (severity) повідомлення, в залежності від цього рівня,

можливо вирішувати куди і як відправляти повідомлення.

У Alertmanager можливо створювати правила та фільтри заглушування (silence) критичних повідомлень.

Інструмент PagerDuty це SaaS платформа яка дозволяє створювати інфраструктури доставки повідомлень про інциденти.

PagerDuty інфраструктура, розгорнута згідно з методом автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем зображена на рисунку 2.35.

PagerDuty сервіс Prometheus alerts сповістив про тестові інциденти. Спираючись на політику оповіщення SRE Escalation Policy, оповіщення про критичний інцидент отримав користувач Володимир Скляр згідно з його контактних методів. У інциденті зазначається його статус – Triggered, назва (з інформацією від якої віртуальної машини отримано сповіщення) та час створення інциденту. Тип інцидентів (Urgency) – високий. Переглядати ці інциденти та їх деталі може команда SRE-team.

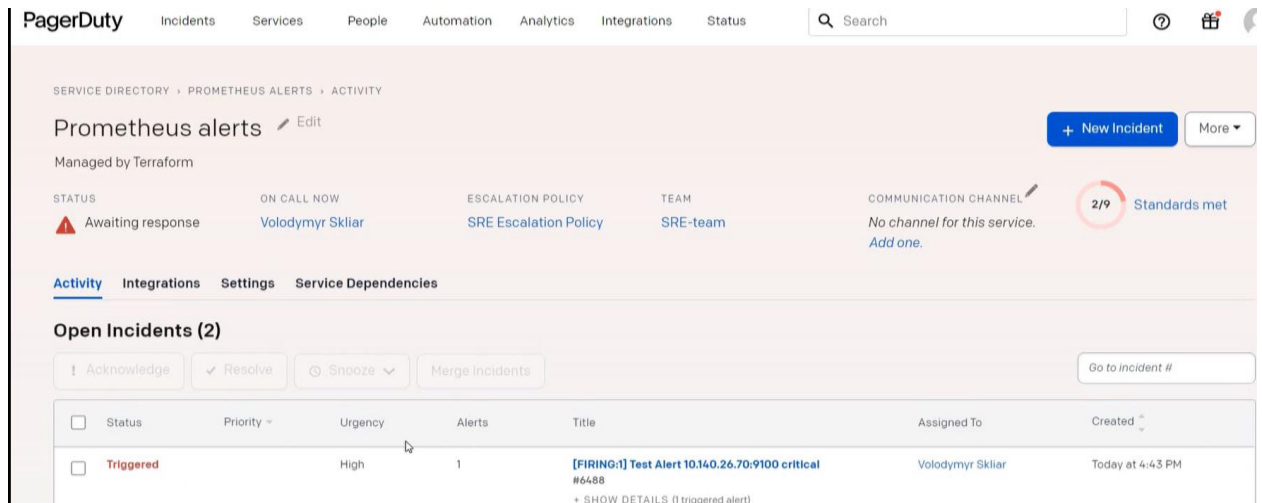


Рисунок 2.35 – Розгорнута PagerDuty інфраструктура

Основними об'єктами PagerDuty на рисунку 2.35 є:

- інцидент – повідомлення про проблему, яке може знаходитися у 4 станах: Triggered – з таким станом повідомлення з'являються у PagerDuty; Acknowledged – користувач PagerDuty побачив повідомлення і прийняв до роботи; Snoozed – користувач провів аналіз проблеми та поставив сповіщення на паузу на деякий час; Resolved – інцидент вирішений, проблема усунута;
- сервіс – сутність у PagerDuty, яка відповідає окремому сервісу для якого налаштована інтеграція;
- політика ескалації – ієрархія отримання повідомлень про інциденти, наприклад, розробник не відреагував на повідомлення протягом 15 хвилин, повідомлення отримує керівник;
- користувач – людина, яка буде отримувати повідомлення базуючись на її контактних методах. Контактних методів повинно бути декілька, наприклад, одразу після отримання інциденту надіслати лист на пошту та надіслати push сповіщення у мобільний додаток, якщо через 5 хвилин інцидент має статус Triggered зателефонувати на мобільний телефон;
- команда – набір користувачів, які відповідають за спільні сервіси та беруть участь у спільних ескалаціях.

ВИСНОВКИ

В ході процесу розробки проведено дослідження існуючих методик автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем, проаналізовані найбільш популярні технології побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем. У процесі аналізу сформовані рольові об'єкти (користувач, адміністратор, власник та ін.). Отримані результати допомогли сформувати нову методику автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем. Звіт виконано згідно [20 – 22].

К основним результатам відносяться:

- 1) Виконано аналіз проблеми автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 2) Проведено дослідження технологій побудови хмарних інфраструктур для систем моніторингу хмарних інформаційних систем.
- 3) Визначено існуючі проблеми та задачі автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 4) Сформовані рольові вимоги об'єкту автоматизації моніторингу інфраструктур.
- 5) Сформовані загальні вимоги автоматизації моніторингу інфраструктур.
- 6) Розроблений метод автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.
- 7) Проведено апробацію розробленого методу автоматизації формування інфраструктур для систем моніторингу хмарних інформаційних систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шаховська Н. Б., Литвин В. В. Проектування інформаційних систем: навчальних посібник. Львів : «Магнолія-2006», 2011, 380 с.
2. Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, Stephen Thorne. The Site Reliability Workbook. O'Reilly Media, Inc. 2018. 508 с.
3. Jennifer Davis, Ryn Daniels. Effective DevOps. O'Reilly Media, Inc. 2016, 410 с.
4. Mikael Krief. Learning DevOps. Packt Publishing Ltd. Birmingham B3 2PB UK. 2019. 489 с.
5. Prometheus documentation. URL: <https://prometheus.io/> (дата звернення: 18.10.2022).
6. PagerDuty documentation. URL: <https://developer.pagerduty.com/> (дата звернення: 18.10.2022).
7. Zabbix documentation. URL: <https://www.zabbix.com/documentation/current/en/> (дата звернення: 19.10.2022).
8. ELK documentation. URL: <https://www.elastic.co/guide/index.html> (дата звернення: 20.10.2022).
9. NewRelic documentation. URL: <https://docs.newrelic.com/> (дата звернення: 21.10.2022).
10. Amazon.com topic. URL: [https://www.britannica.com/topic/Amazon com](https://www.britannica.com/topic/Amazon-com) (дата звернення: 21.10.2022).
11. Gitlab documentation. URL: <https://docs.gitlab.com/ee/> (дата звернення: 22.10.2022).
12. Michael Heap. Ansible From Beginner to Pro. 2016. 170 с.
13. James Turnbull. The Terraform Book, 2019, 86 с.
14. Terraform documentation. URL: <https://developer.hashicorp.com/terraform/language> (дата звернення: 23.10.2022).

15. Buch G, Rambo D., YAkobson A.. The Unified Modeling Language: Users Guide / G. Buch. – М.: «DMK Press », 2007. – 496 s.
16. UserStories. URL: <https://www.atlassian.com/agile/projectmanagement/-user-stories> (дата звернення 25.10.2022)
17. Sander Rossel. Continuous Integration, Delivery, and Deployment. Packt Publishing Ltd. Birmingham B3 2PB UK. 2017. 541 с.
18. Yan Kurniawan. Ansible for AWS, 2016, 106 с.
19. Joe Baron. AWS Certified Solutions Architect Official Study Guide, 2017, 487 с.
20. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.
21. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 20 с.
22. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи (для студентів усіх форм навчання другого (магістерського) рівня програми "Інформаційні управляючі системи та технології") / Упоряд.:Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2021. – 30с.