УДК 004.45

В.В. Заяц
Л. А. Савицька

СИСТЕМА РОЗГОРТАННЯ ТА МОНІТОРИНГУ ПРИКЛАДНИХ СЕРВІСІВ ІЗ ІНТЕГРАЦІЄЮ DEVOPS-ПРАКТИК

Вінницький національний технічний університет

Анотація

У роботі розглянуто створення системи автоматизованого розгортання та моніторингу прикладних сервісів із інтеграцією DevOps-практик. Система забезпечує автоматизацію процесів конфігурації серверів, розгортання контейнеризованих застосунків, організацію CI/CD-конвеєра та безперервний моніторинг інфраструктури. Розроблена система підвищує ефективність розгортання, зменшує вплив людського фактора та забезпечує надійність серверних сервісів у продукційному середовищі.

Ключові слова: автоматизація, розгортання, моніторинг, CI/CD, DevOps, Terraform, Ansible, Kubernetes, Prometheus.

Abstract

The work considers the creation of a system of automated deployment and monitoring of applied services with the integration of the Devops practitioner. The system provides automation of server configuration processes, the deployment of containerized applications, the CI/CD conveyor organization and continuous infrastructure monitoring. The developed system increases the efficiency of deployment, reduces the impact of the human factor and ensures the reliability of server services in the product environment.

Keywords: automation, deployment, monitoring, CI/CD, Devops, Terraform, Ansible, Kubernetes, Prometheus.

Вступ

Стрімкий розвиток інформаційних технологій, зокрема хмарних обчислень, контейнеризації та мікросервісної архітектури, докорінно змінив підходи до створення, розгортання та супроводу програмних продуктів. Сучасні корпоративні системи все частіше складаються з десятків або навіть сотень взаємопов’язаних сервісів, які потребують узгодженого керування, гнучкого масштабування та безперебійної роботи. За таких умов класичні підходи до адміністрування, що передбачають ручне налаштування серверів і сервісів, втрачають свою ефективність, оскільки не можуть забезпечити необхідну швидкість розгортання, стабільність і повторюваність середовищ. Це створює потребу у впровадженні автоматизованих рішень, здатних мінімізувати людський фактор, забезпечити стабільність оновлень і гарантувати контроль якості програмного продукту на кожному етапі його життєвого циклу.Одним із найбільш ефективних підходів, що відповідає цим вимогам, є методологія DevOps (Development + Operations), яка поєднує процеси розробки програмного забезпечення та його експлуатації в єдиний безперервний цикл.

Постановка задачі дослідження

У межах дослідження метою є розробка комплексної системи автоматизованого розгортання та моніторингу прикладних сервісів із використанням DevOps-практик, яка забезпечує повний цикл управління життєвим циклом прикладних сервісів. Для досягнення поставленої мети необхідно провести аналітичний огляд сучасних DevOps-технологій, визначити їхні сильні сторони, принципи інтеграції та можливості спільного використання. Спроєктувати архітектуру системи розгортання. Реалізувати керування інфраструктурою як кодом із використанням Terraform для створення та підтримання серверних ресурсів. Забезпечити автоматизацію налаштувань середовища та підготовку серверних ресурсів за допомогою Ansible. Виконати контейнеризацію застосунків з використанням Docker та налаштувати оркестрацію контейнерів у середовищі Kubernetes.. Реалізувати CI/CD-конвеєр у Jenkins, який автоматизує процеси тестування, збирання та розгортання застосунків при зміні вихідного коду. Розробити систему моніторингу на основі Prometheus і Grafana для збору, зберігання та візуалізації метрик серверів і контейнерів у реальному часі. Провести тестування та оцінку ефективності системи, визначити показники продуктивності, стабільності та масштабованості.

Результати дослідження

У процесі виконання дослідження було розроблено повнофункціональну систему, що реалізує автоматизацію основних етапів життєвого циклу серверних сервісів — створення, налаштування, розгортання, моніторинг та візуалізацію результатів. Отримане рішення поєднує інструменти DevOps-екосистеми в єдину архітектуру, забезпечуючи узгоджену роботу всіх її компонентів.

На першому етапі реалізації було розроблено інфраструктурні модулі Terraform, які описують усі необхідні ресурси середовища: віртуальні машини, мережеві інтерфейси, підмережі, сховища даних та правила доступу. Це дало змогу автоматизувати процес створення інфраструктури та усунути залежність від ручних дій адміністратора. Використання принципу “інфраструктура як код” дозволило фіксувати зміни у системі контролю версій, відтворювати середовище у будь-який момент часу та забезпечити повну відповідність між різними інстанціями серверів. [1,5]

Після опису та розгортки основних сервених ресурсів було інтегровано систему Ansible для автоматизації налаштування конфігурацій серверів. З її допомогою реалізовано встановлення базових системних пакетів, налаштування користувачів, конфігурацію мережевих сервісів і підготовку контейнерного середовища. Ansible забезпечує декларативний підхід, коли описується бажаний стан системи, а інструмент автоматично приводить її у відповідність до цього стану. [6] Це значно підвищує надійність та узгодженість налаштувань у багатосерверному середовищі.

Для ізоляції та портативності застосунків використано технологію Docker, яка дозволяє створювати контейнери з усіма залежностями програмного забезпечення. Це забезпечило можливість однакового розгортання сервісів у будь-якому середовищі — локальному, тестовому чи продукційному. Для масштабування контейнерів і підтримки їх безперервної роботи впроваджено Kubernetes, який виконує функції оркестрації, автоматичного балансування навантаження, перезапуску збоїв і розподілу ресурсів між вузлами кластеру. [3] Завдяки цьому система здатна стабільно працювати навіть при збільшенні кількості сервісів чи навантаження.

Щоб автоматизувати процеси збирання, тестування та розгортання коду прикладного сервісу було створено CI/CD-конвеєр у Jenkins. Він інтегрується з системою контролю версій Git та реагує на зміну коду, автоматично виконуючи збирання контейнерів, запуск тестів і оновлення розгорнутих сервісів. Такий підхід скорочує час між написанням коду та його доставкою у продакшн і знижує ризики, пов’язані з людськими помилками під час розгортання.

Підсистему моніторингу та візуаліазції було реалізовано за допомогою Prometheus і Grafana. Prometheus відповідає за збір метрик із серверів, контейнерів і кластерів Kubernetes, зберігаючи їх у часових рядах, що дозволяє виконувати аналітику у динаміці. Grafana забезпечує зручну візуалізацію даних, створення інтерактивних дашбордів і сповіщень у разі виявлення відхилень. Це забезпечує безперервний контроль стану інфраструктури, дає змогу швидко виявляти проблеми та реагувати на них у реальному часі. [4]

У процесі тестування система продемонструвала високу ефективність та стабільність. Завдяки використанню контейнеризації та IaC забезпечено відтворюваність середовища та швидке масштабування. Автоматизація процесів конфігурації та розгортання скоротила час введення нових сервісів у експлуатацію на 60–70%, а впровадження CI/CD-конвеєра мінімізувало ризик збоїв у процесі оновлення. Система моніторингу дозволила оперативно виявляти перевантаження ресурсів, падіння контейнерів або зниження продуктивності.

Висновки

У результаті проведеного дослідження було створено комплексну систему автоматизованого розгортання та моніторингу прикладних сервісів із інтеграцією DevOps-практик. Розроблене рішення поєднує ключові принципи сучасного підходу до управління інфраструктурою — автоматизацію, контейнеризацію, безперервну інтеграцію та спостережуваність. У межах роботи реалізовано повний цикл DevOps-процесів — від опису інфраструктури як коду до побудови CI/CD-конвеєра та централізованого моніторингу системи. Практичне значення розробленої системи полягає у можливості її застосування в IT-відділах підприємств, хмарних інфраструктурах, центрах обробки даних і навчальних закладах для демонстрації сучасних підходів до автоматизації. Виконане дослідження підтвердило доцільність впровадження DevOps-практик у процес управління прикладними сервісами. Розроблена система є гнучким, надійним і сучасним рішенням, що дозволяє ефективно керувати інфраструктурою, прискорювати цикл розробки та підвищувати загальну якість інформаційних систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Азаров О. Д. Комп'ютерні мережі / О. Д. Азаров, С. М. Захарченко, О. В. Кадук та ін. — Вінниця : ВНТУ, 2013. — 370 с. ISBN 978-966-641-543-4

2. Devops Principles | Atlassian [Електронний ресурс] - Режим доступу до ресурсу: https://www.atlassian.com/devops

3. Cluster Architecture [Електронний ресурс] - Режим доступу до ресурсу: https://kubernetes.io/docs/concepts/architecture/

4.Monitoring and Visualization for Release 1.2 [Електронний ресурс] - Режим доступу до ресурсу: https://docs.oracle.com/en/operating-systems/olcne/1.2/monitor/

5. What is Infrastructure as Code? - IaC Explained - AWS [Електронний ресурс] - Режим доступу до ресурсу: https://aws.amazon.com/what-is/iac/

6. What is Configuration Management? - Software Configuration Management Explained - AWS [Електронний ресурс] - Режим доступу до ресурсу: https://aws.amazon.com/what-is/configuration-management/

Заяц Владислав Валерійович – студент групи 1КІ-24м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: vladyslavzayats@gmail.com

Савицька Людмила Анатоліївна – к.т.н., доцент кафедри обчислювальної техніки, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: savytska.liudmyla@vntu.edu.ua

Zayats Vladyslav Valeriyovych – student of group 1ki-24m, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnitsa, e-mail: vladyslavzayats@gmail.com

Savitska Lyudmila Anatoliivna – Ph.D., Associate Professor of the Department of Computer Engineering, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: savytska.liudmyla@vntu.edu.uau.ua