УДК 004.415.052

МЕТОДОЛОГІЯ CHAOS ENGINEERING У DEVOPS ЯК ЗАСІБ МОДЕЛЮВАННЯ ВІДМОВ У CI/CD ДЛЯ ПІДВИЩЕННЯ НАДІЙНОСТІ СИСТЕМ

Матвєєв М.С.
e-mail: mykhailo.matvieiev@nure.ua
Харківський національний університет радіоелектроніки, каф. КІТС,
м. Харків, Україна

The present paper explores the potential of Chaos Engineering as a methodology for enhancing the reliability of software systems within the DevOps environment. Chaos Engineering facilitates the emulation of arbitrary failures within Continuous Integration/Continuous Delivery processes, thereby aiding in the identification of latent vulnerabilities prior to their actualisation. The paper delves into the foundational tenets of Chaos Engineering, examines prevalent instruments (Gremlin, LitmusChaos, Chaos Monkey), and investigates their implementation in cloud and containerised environments.

У сучасному цифровому світі, де швидкість розробки та впровадження програмного забезпечення має вирішальне значення, забезпечення надійності систем набуває особливої важливості. Розвиток підходу DevOps сприяв автоматизації процесів створення, тестування та впровадження програмних рішень, що значно скоротило цикл випуску оновлень. Навіть у разі використання практик неперервної інтеграції та доставки (CI/CD) існують ризики виникнення ситуацій відмов, які спричинені невизначеними факторами, такими як помилки конфігурації, неочікувані пікові навантаження або збої у мережевій інфраструктурі. В таких умовах велике значення має Chaos Engineering – методологія, яка спрямована на виявлення потенційних вразливостей системи шляхом навмисного створення збоїв в контрольованих умовах.

Метою дослідження стало обґрунтування доцільності впровадження Chaos Engineering у процеси DevOps та CI/CD як інструменту забезпечення надійності програмних систем та підвищення їхньої готовності до непередбачених збоїв. У процесі роботи було проведено аналіз основних принципів Chaos Engineering, визначено їхню роль у формуванні стратегії надійності, а також досліджено сучасні інструменти, які дозволяють автоматизувати проведення таких експериментів. Важливим завданням стало вивчення реального досвіду компаній-лідерів у цій сфері, зокрема Google та Netflix, які вже активно впроваджують практики Chaos Engineering у своїх CI/CD-конвеєрах.

Chaos Engineering ґрунтується на основних принципах, які забезпечують ефективне тестування надійності програмних систем у контексті DevOps.

Перший принцип полягає у формулюванні гіпотези про стійкість системи, що передбачає розроблення обґрунтованих припущень щодо

реакції системи на різні види відмов. Цей процес ґрунтується на глибокому аналізі архітектури, ідентифікації критичних компонентів, моделюванні потенційних ризиків та визначенні ключових показників продуктивності, таких як час відновлення (MTTR), середній час безвідмовної роботи (MTBF) та рівень деградації сервісів під час збоїв. Формулювання гіпотези має велике значення, оскільки безпорядне впровадження змін без чіткого розуміння очікуваних результатів може призвести до дестабілізації системи і не принести корисних інсайтів. Наприклад, при тестуванні мікросервісної архітектури важливо визначити критичні сервіси для підтримання загальної функціональності, врахувати наявність резервних механізмів у деяких сервісів, а також визначити терпимість короткочасних перебоїв у роботі без суттєвого впливу на користувачів. Дослідження Google SRE показують, що точне формулювання гіпотез перед проведенням експериментів може допомогти зменшити кількість несподіваних відмов у виробничому середовищі на 60%, що свідчить про важливість цього етапу[1].

Другим принципом є реалістичне моделювання відмов, спрямоване на відтворення небажаних сценаріїв у контрольованих умовах. Це може включати свідому втрату зв'язку, перевантаження обчислювальних ресурсів, порушення роботи балансувальників навантаження або відмову окремих сервісів у мікросервісній архітектурі тобто, що ефективність Chaos Engineering визначається реалістичністю тестових умов і моделювання випадкових збоїв без врахування реальних особливостей інфраструктури може призвести до отримання неточних результатів та недостатнього відображення справжніх ризиків [2].

Третій значущий аспект – це автоматизація експериментів, включаючи використання спеціалізованих інструментів для впровадження контрольованих відмов. Сучасні інструменти, такі як Gremlin, LitmusChaos та Chaos Monkey, надають можливість проведення експериментів з різним рівнем складності, починаючи від простого переривання мережевих з'єднань до масштабних сценаріїв руйнування цілих компонентів. Наприклад, підходи, впроваджені у Netflix, такі як Chaos Monkey, який випадковим чином вимикає сервіси у хмарному середовищі, сприяють виявленню слабких місць у високонавантажених розподілених системах [3]. У свою чергу, LitmusChaos спеціалізується на проведенні тестів Kubernetes-кластерів та забезпечує інтеграцію з пайплайнами CI/CD, що робить його ефективним інструментом для тестування надійності у контейнеризованих середовищах [3].

Останнім етапом є проведення аналізу отриманих даних, що забезпечує зворотний зв'язок для подальшого вдосконалення системи. Після здійснення експериментів важливо провести аналіз метрик продуктивності, часу відновлення, рівня деградації сервісів та відповідність фактичної поведінки системи очікуваним результатам.

Необхідно звертати особливу увагу на виявлення прихованих залежностей між сервісами. Результати дослідження в 2022 році показали, що 73% критичних відмов у мікросервісних системах виникають внаслідок неочікуваних взаємодій між на перший погляд незалежними компонентами.

Інтеграція Chaos Engineering у процеси CI/CD сприяє підвищенню стійкості розгортання оновлень та запобігає виникненню аварійних ситуацій у виробничому середовищі. Одним з важливих варіантів використання цього підходу є проведення тестування стійкості під час випуску нових версій програмного забезпечення. Ця практика охоплює проведення контрольованих експериментів під час оновлення версій програмного забезпечення, які дозволяють виявити проблеми ще до того, як вони з'являться у середовищі розробки. Наприклад, в реаліях відділу інженерії надійності сайту у Google активно використовується метод "канаркових випусків" ("canary releases"), при якому невеликий відсоток користувачів отримує нову версію програмного забезпечення, і метод Chaos Engineering використовується для перевірки її стабільності.

Таким чином, у контексті DevOps та CI/CD методологія Chaos Engineering грає ключову роль у забезпеченні надійності програмних продуктів, дозволяючи організаціям проактивно ідентифікувати слабкі місця на етапах розробки та тестування. Результати проведеного дослідження свідчать, що інтеграція цього підходу сприяє не лише виявленню вразливостей, а й створенню більш стійких архітектур, що дозволяють ефективно реагувати на реальні збої у виробничому середовищі. І також те, що компанії, які впровадили автоматизовані хаотичні тести у свої пайплайни CI/CD, змогли знизити середній час відновлення після інцидентів на 40%, а точне формулювання гіпотез перед експериментами дозволило зменшити кількість неочікуваних відмов на етапі продакшену на 60%. Отримані результати дозволяють рекомендувати активне застосування Chaos Engineering у процесах неперервної інтеграції та доставки, особливо у компаніях, які працюють із хмарними та розподіленими системами, що потребують високої відмовостійкості.

Список використаних джерел:

1. Hochstein L., Basiri A., Blohowiak A., Rosenthal C. Chaos Engineering // IEEE Software. 2016. №33(3). С. 35-41. DOI: 10.1109/MS.2016.60

2. Assad M., Meiklejohn C., Miller H., Krusche S. Can My Microservice Tolerate an Unreliable Database? Resilience Testing with Fault Injection and Visualization // arXiv preprint arXiv:2404.01886. 2024. DOI: 10.48550/arXiv.2404.01886

3. Challa V.N.S.K. Chaos Engineering for Building Resilient Distributed Systems // International Journal of Science and Research (IJSR). 2020. Т. 9, №3. С. 177-181. DOI: 10.21275/SR24716231253