УДК 004.415

МОЖЛИВОСТІ ТА ПЕРЕВАГИ КОНТЕЙНЕРИЗАЦІЇ ПІД ЧАС ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мирошниченко С.А.
Науковий керівник – ст.викл. Олійник О.В.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
тел.: +38(098) 050-89-36, e-mail: serhii.myroshnychenko@nure.ua

The text discusses the benefits of using Docker, a containerization tool, in a microservice architecture. Docker allows for automated creation of isolated containers, each with all necessary components for stable operation on any platform. The Docker kernel uses client-server technology for easy interaction between containers. The microservice architecture divides the project into separate parts, each run in a separate container, and Kubernetes is used for orchestration. Logging and monitoring systems are connected to track operations and resource consumption. Docker speeds up development and supports highly loaded systems.

Набуття популярності мікросервісної архітектури та потреба в безперервній інтеграції й доставці програмного коду, призвела до створення сучасного інструменту контейнеризації Docker [1]. Даний інструмент являється автоматизованим механізмом для роботи з віртуальними контейнерами. Кожний контейнер являє собою окремий ізольований простір, котрий будується, опираючись на відповідний Docker образ. В результаті контейнер буде містити всі необхідні компоненти, котрі забезпечать стабільну та передбачувану роботу на будь-якій платформі. Хоч Docker підтримує багато платформ, найбільшої популярності набув саме Linux, безкоштовний та легкий в користуванні.

Процес віртуалізації виконується саме на рівні операційної системи, в результаті чого, всю основну роботу покладено на її ядро та ресурси, котрими Docker дозволяє управляти. Особливо корисною являється можливість розподілу ресурсів між всіма контейнерами, так як дуже часто на різні контейнери припадає різне навантаження. Правильний розподіл дозволить в найкритичніші моменти розвантажити систему та стабілізувати ситуацію в найбільш навантажених контейнерах. Така легка взаємодія між контейнерами доступна за рахунок клієнт серверної технології, що лежить в основі ядра Docker (рис. 1) [2].

Головними складовими ядра являються сервер, REST API та клієнт. Сервер відповідає за створення та ініціалізацію Docker-daemon, котрий забезпечує управління образами та відповідними контейнерами. REST API являє собою інструмент взаємодії клієнта та сервера. Клієнт – компонент взаємодії інструмента з користувачем, котрий представлений у вигляді команд.

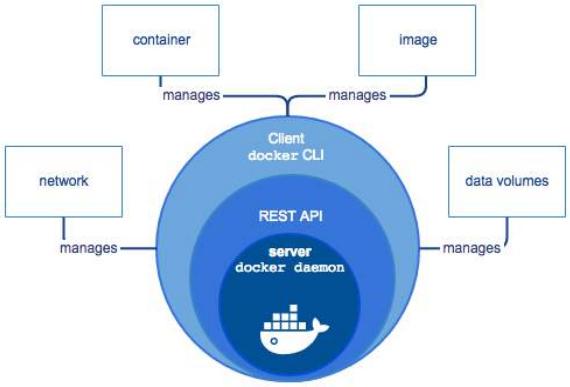

Рисунок 1 – Внутрішня структура Docker

Опираючись на вищесказане, була обрана мікросервісна архітектура, в основі якої лежить Docker контейнер [3]. Під час її проектування, необхідно зауважити, що кожний мікросервіс повинен бути невеликого розміру, та максимально незалежним від інших. Для цього, весь проект був розділений на окремі частини, кожна з яких запускалася в окремому контейнері. Так як система складалася з великої кількості окремих мікросервісів, постало питання стосовно її оркестрації. Найбільш популярним в наш час виявився Kubernetes, він і був обраний в ролі інструмента для автоматизації та розгортання всієї системи.

Очевидно, що за кожним мікросервісом потрібно наглядати, для цього було підключено системи логування та моніторингу. Система логування дозволить відстежувати всі операції, котрі виконував мікросервіс. При появі будь-яких проблем, дана система дозволить швидко знайти їх корінь, та, в результаті чого, швидко їх виправити. Логи необхідно створювати під час роботи кожного окремого контейнеру, для постійного аналізу стану відповідного мікросервісу. Система моніторингу дозволить відстежувати навантаження на сервер та аналізувати поточні витрати ресурсів на окремий контейнер. В результаті, Docker виявився дуже важливим інструментом. До його переваг можна віднести пришвидшення процесу розробки, зрозумілий інтерфейс, зручну систему моніторингу та простоту процесу масштабування. Він виявився саме тим інструментом, котрий дозволив швидко та зручно підтримувати високонавантажені системи.

Список використаних джерел:

1. Docker Docs. (2023, 27 лютого). How to build, share, and run applications. https://docs.docker.com

2. Docker Docs. (2023, 27 лютого). Docker overview. https://docs.docker.com.xy2401.com/engine/docker-overview/

3. Вікіпедія (2023, 27 лютого). Мікросервіси. https://uk.wikipedia.org/wiki/Мікросервіси