ЗАСТОСУВАННЯ DOCKER ТА JENKINS ДЛЯ СТВОРЕННЯ СЕРЕДОВИЩА РОЗГОРТАННЯ ТА ЗАПУСКУ ЗАСТОСУНКІВ

Котоміна К.В., Колтун Ю.М.
Харківський національний університет радіоелектроніки, Харків, Україна

На сьогоднішній день одними з найпопулярніших інструментів для розробки і розгортання програмних застосунків є платформа контейнеризації Docker і сервер автоматизації з відкритим вихідним кодом Jenkins. Docker автоматизує процеси створення, запуску і управління застосунками в легких контейнерах, які можна переносити між різними системами та інфраструктурами. Для цього він містить все необхідне: код, середовище виконання, системні інструменти, системні бібліотеки та параметри [1]. Тобто розробники більше не замислюються над тим, у якому середовищі функціонуватиме застосунок і чи будуть у цьому середовищі необхідні для тестування опції та залежності. Jenkins являє собою Java-застосунок, який використовується для організації процесів безперервної інтеграції (CI) та безперервної доставки (CD) програмних застосунків. Jenkins надає сотні плагінів, які дозволяють розробникам автоматично збирати, тестувати та розгортати застосунки, значно прискорюючи їх розробку і запуск з мінімальним впливом людського фактору [2].

Таким чином, застосування Jenkins разом з платформою Docker є потужним середовищем розробки застосунків та підвищує ефективність процесів автоматизації їх розгортання, запуску і управління, що є актуальною задачею.

Метою доповіді є опрацювання методики розробки, розгортання і запуску програмного застосунку інструментами Docker та Jenkins. Також для реалізації цього проєкту були застосовані Java-фреймворки для створення масштабованих застосунків із модульною архітектурою та управлінням залежностями, такі як Spring Boot та Spring Data. Для зберігання даних використана об'єктно-реляційна СУБД PostgreSQL, а автоматичною збіркою цього проєкту опікувався Apache Maven.

У доповіді показано, що спочатку треба завантажити image (образ) Jenkins із Docker hub [3]. Після цього запускаємо контейнер за допомогою команди <docker run>, вказавши необхідні опції: порти, назву контейнера та сам образ.

Однак, якщо крім «чистого» Jenkins-оточення потрібно використовувати застосункові програми, тоді треба створити Dockerfile.

Зокрема створення Dockerfile дозволило автоматично завантажити плагіни Jenkins для нашого застосунку. Для цього у кореневій папці, в якій розташований Dockerfile, був створений файл plugins.txt, у якому описуються всі необхідні плагіни у форматі <назва плагіну>:<версія>. Команди, які потрібно застосуни до Dockerfile, наступні:

COPY/plugins.txt/usr/share/jenkins/ref/plugins.txt

RUN/usr/local/bin/install-plugins.sh</usr/share/jenkins/ref/plugins.txt

Далі потрібно створити Docker-образ Tomcat і PostgreSQL для розгортання і зберігання даних застосунку, що був автоматично зібраний інструментом Maven-плагіном, що встановлений в Jenkins. Тут треба

враховувати, що всі контейнери мають утворювати зв'язану систему для контролю за збірками та розгортанням.

Для цього необхідно прокинути порти між контейнерами Tomcat і PostgreSQL, а Jenkins підключити за допомогою загального сховища volume (загальний диск, папка), що є засобом зберігання та розподілу даних між контейнерами, незалежно від їх життєвого циклу.

Для цього при запуску Jenkins-контейнера використовуємо опцію – volumes-from=tomcat:rw, а при запуску Tomcat опцію –v=tomcat-data:/usr/local/tomcat/webapps:rw. Таким чином volume tomcat-data стає доступною в Jenkins [4].

Зв’язати по мережі Tomcat і PostgreSQL для отримання даних дозволяє опція –link, яка використовується у команді, наприклад, при запуску Tomcat-контейнера:

docker run –it -p 8080:8080 --name tomcat -v tomcat-data:/usr/local/tomcat/webapps:rw --link postgres:postgres -d tomcat:latest

Звідси у підсумку показано, що можна запускати Jenkins-контейнер використовуючи команду [4]:

docker run -p 8888:8080 -p 50000:50000 --name=jenkins-master --mount source=jenkins-log,target=/var/log/jenkins --mount source=jenkins-data,target=/var/jenkins_home --volumes-from=tomcat:rw -d jenkins-artteam

Зокрема можна бачити, що у вищенаведеній команді створюються volumes для зберігання і відновлення даних. Jenkins після запуску буде доступний на http://localhost:8888/.

В результаті проведеного аналізу і розробки було забезпечено автоматизований контроль за збіркою та розгортанням програмного застосунку за рахунок створення узгодженої системи контейнерів Docker. Використання Jenkins дозволило створити середовище тестування застосунку, а користувачам цієї системи були визначені умови для самостійного прописування етапів, які необхідно пройти проєкту перед його запуском, як повноцінного застосунку.

Список літератури

1. Adrian Mouat Using Docker: Developing and Deploying Software with Containers. Sebastopol, California, USA: O'Reilly Media. – 2016. – p. 358.

2. Jenkins User Documentation [Електронний ресурс] / Jenkins. – Доступ здійснено 08.04.2026. – Режим доступу до ресурсу: https://www.jenkins.io/doc/

3. Docker Hardened Images [Електронний ресурс]. – Режим доступа: https://hub.docker.com

4. Lieber M. Home CICD: Docker, Jenkins and Nexus3 [Електронний ресурс] / Mike Lieber // Habr. – 2022. – Режим доступу до ресурсу: https://habr.com/ru/articles/677142/.