Торопов Д.Ю. Анализ и проведение эксперимента по тестированию веб-приложения с использованием контейнеров // Международный журнал информационных технологий и энергоэффективности. – 2024. – Т. 9 № 7(45) с. 168–172

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.05

АНАЛИЗ И ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТА ПО ТЕСТИРОВАНИЮ ВЕБ-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ КОНТЕЙНЕРОВ

Торопов Д.Ю.
ФГАОУ ВО "НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ (ИТМО)», Санкт-Петербург, Россия (197101, город Санкт-Петербург, Кронверкский пр-кт, д. 49 литер а), e-mail: toropov85@yandex.ru

Настоящая статья представляет собой обзор исследований и экспериментальных результатов, связанных с использованием Docker, GitHub Actions для автоматизации процессов CI/CD в разработке веб-приложений. Она акцентирует внимание на актуальности этих инструментов для повышения эффективности и автоматизации разработки ПО, а также на обнаруженных преимуществах и проблемах при их использовании.

Ключевые слова: Docker, контейнер, веб-приложение, CI/CD, GitHub Actions.

ANALYSIS AND CONDUCTING AN EXPERIMENT TO TEST A WEB APPLICATION USING CONTAINERS

Toropov D.Y.
NATIONAL RESEARCH UNIVERSITY OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS (ITMO), St. Petersburg, Russia (197101, St. Petersburg, Kronverkskiy pr-kt, 49,lit.a), e-mail: toropov85@yandex.ru

This article is a review of research and experimental results related to the use of Docker, GitHub Actions for automating CI/CD processes in web application development. It focuses on the relevance of these tools for improving the efficiency and automation of software development, as well as the benefits and problems discovered in their use.

Keywords: Docker, container, web application, CI/CD, GitHub Actions.

Введение

При современном развитии информационных технологий, где актуальна скорость прогресса, разработчики веб-приложений сталкиваются с возросшими требованиями к эффективности и надежности своих продуктов.

Обеспечение бесперебойной работы в условиях высоких нагрузок становится ключевым элементом успеха. Пользователи, сталкиваясь с неудовлетворительной производительностью, имеют тенденцию отказываться от использования приложений в пользу более эффективных альтернатив.

Нагрузочное тестирование играет важную роль в обеспечении стабильной работы программного обеспечения, позволяя разработчикам оценить его работоспособность и выявить узкие места производительности.

С учетом быстрого развития веб-технологий и микросервисной архитектуры, а также широкого использования контейнеризации, в том числе с использованием Docker, крайне важно провести глубокий анализ влияния этих нововведений на процессы тестирования веб-приложений, а также на их производительность и надежность. Важно исследовать влияние CI/CD на тестирование и производительность, а также влияние различных конфигураций оборудования на производительность веб-приложений.

Анализ литературы

В современном программном обеспечении принято использовать методологию непрерывной интеграции и непрерывной доставки (CI/CD), что позволяет значительно сократить время между внесением изменений в исходный код и публикацией обновлений на веб-сайтах и в приложениях [1]. Этот процесс основывается на автоматизации всех этапов, начиная с тестирования кода и заканчивая его автоматическим развертыванием.

В наше время многие разработчики используют такие инструменты как Docker и GitLab для автоматизации процесса разработки и развертывания [2]. Например, Docker обеспечивает совместимость приложений в разных средах благодаря упаковке приложений и их зависимостей в контейнеры.

GitLab предоставляет средства управления исходным кодом, а также возможность настройки процессов CI и CD [3]. Также можно использовать такие инструменты как AWS CodeDeploy для развертывания приложений на инфраструктуре Amazon Web Services, a GitHub Actions для автоматизации процессов CI/CD [4].

Однако следует отметить, что в процессе CI/CD могут возникнуть определенные сложности [5]. Например, изменения в условиях окружения или внешних сервисах могут привести к неожиданным результатам в сборках. Также существует необходимость в дальнейших исследованиях по управлению зависимостями между действиями в сценариях сборки, чтобы обеспечить стабильность и надежность процесса CI/CD.

Результаты нагрузочного тестирования

В рамках исследования разработано простое веб-приложение на Node.js с использованием Express, предназначенное для проведения нагрузочного тестирования. Для автоматизации процессов CI/CD и интеграции сборки, тестирования и развертывания Docker образа был разработан и внедрен GitHub Actions workflow.

При каждом обновлении ветки master workflow автоматически клонирует репозиторий, устанавливает Node.js, выполняет установку зависимостей, сбрасывает кэш Docker Hub для пере аутентификации, а затем осуществляет сборку и отправку Docker-образа с прописанным тегом. Этот процесс позволяет обеспечить стабильность и автоматизацию процессов разработки и развертывания.

После успешного развертывания Docker-образа была проведена аутентификация в приложении Desktop Docker на двух персональных компьютерах HP и Huawei. Затем на каждом компьютере был запущен Docker образ и проведено нагрузочное тестирование с использованием Apache Benchmark. Тестирование проводилось над веб-приложением с контейнерами и без их применения в двух итерациях на каждом персональном компьютере.

В первом тесте, без Docker-контейнеризации, сервер обработал 1000 запросов за 0.709 секунды, что соответствует средней скорости 1409.68 запросов в секунду. Время ответа составило около 7.094 миллисекунд, а общий объем переданных данных - 213000 байт.

Во втором тесте с уровнем параллели 100, сервер обработал миллион запросов за 473.97 секунды, что соответствует средней скорости 2109.84 запросов в секунду. Время ответа составило 47.397 миллисекунд, а общий объем переданных данных - 213000000 байт.

С применением Docker-контейнеризации в первом тесте сервер обработал 1000 запросов за 1.124 секунды, что соответствует средней скорости 889.50 запросов в секунду. Время ответа составило около 11.242 миллисекунд, а общий объем переданных данных - 218000 байт.

Во втором тесте с уровнем параллели 100, сервер обработал миллион запросов за 889.76 секунды, что соответствует средней скорости 1123.90 запросов в секунду. Время ответа составило 88.976 миллисекунд, а общий объем переданных данных - 218000000 байт.

Анализ результатов показывает, что производительность сервера немного ухудшилась с применением Docker-контейнеризации при уровне параллели 10, но при уровне параллели 100 произошло улучшение.

Время ответа на запрос возросло с применением Docker, что может указывать на более высокое время ответа на отдельные запросы. Общий объем переданных данных увеличился, но остался в пределах сопоставимых значений.

Таблица сравнения тестирований на ноутбуке HP представлена в Таблице 1.

Таблица 1 – Результаты нагрузочного тестирования на ноутбуке HP

Метрика	**Без Docker (параллель= 10)**	**Без Docker (параллель= 100)**	**С применением Docker (параллель= 10)**	**С применением Docker (параллель= 100)**
Время тестирования (с)	0.709	473.970	1.124	889.760
Запросы в секунду (запрос/с)	1409.68	2109.84	889.50	1123.90
Время на запрос (мс)	7.094	47.397	11.242	88.976
Объем данных (общий) (байт)	213000	213000000	218000	218000000
Объем данных (HTML) (байт)	14000	14000000	18000	18000000

В ходе эксперимента на ноутбуке Huawei было проведено сравнение производительности приложения, запущенного в разных сценариях. В первом сценарии, без использования Docker, приложение обрабатывало 1,000 запросов с параллелизмом 10, достигая средней производительности в 2,171.07 запросов в секунду. Среднее время обработки запроса составляло 4.606 миллисекунд. Во втором сценарии, также без Docker, приложение обрабатывало 1,000,000 запросов с параллелизмом 100, достигая средней производительности 3,960.80 запросов в секунду и среднего времени обработки запроса 25.247 миллисекунд.

При использовании Docker производительность приложения в обоих тестах была ниже: при обработке 1,000 запросов с параллелизмом 10 средняя производительность составила 831.62 запроса в секунду, среднее время ответа на запрос - 12.025 миллисекунд, и при обработке

1,000,000 запросов с параллелизмом 100 средняя производительность составила 626.74 запроса в секунду, среднее время ответа на запрос - 159.555 миллисекунд.

Общий объем переданных данных рос пропорционально количеству запросов, и в обоих тестах не было обнаружено отклоненных запросов. Результаты позволяют сделать вывод о масштабируемости приложения и его эффективности при обработке высоких нагрузок в различных сценариях.

Результаты тестирования на ноутбуке Huawei представлены в таблице 2.

Таблица 2 – Результаты нагрузочного тестирования на ноутбуке Huawei

Метрика	**Без Docker (параллель= 10)**	**Без Docker (параллель= 100)**	**С применением Docker (параллель= 10)**	**С применением Docker (параллель= 100)**
Время тестирования (с)	0.461	252.474	1.202	1595.553
Запросы в секунду (запрос/с)	2171.07	3960.80	831.62	626.74
Время на запрос (мс)	4.606	25.247	12.025	159.555
Объем данных (общий) (байт)	218000	218000000	218000	218000000
Объем данных (HTML) (байт)	18000	18000000	18000	18000000

Сравнительный анализ показал, что влияние Docker на производительность зависит от характеристик оборудования и настроек системы. Например, на ноутбуке HP наблюдалось уменьшение времени тестирования с использованием Docker, однако запросы в секунду снижались. В то время как на ноутбуке Huawei, при 100 параллельных запросах, наблюдалось увеличение времени тестирования и объема переданных данных. Эти различия могут быть обусловлены пропускной способностью оборудования, настройками системы и влиянием Docker на производительность. Таким образом, результаты подчеркивают необходимость детального анализа и тестирования при использовании Docker для определения оптимальных условий работы данного программного инструмента.

Список литературы

1. Коняева О.С., Системы контроля версий в процессе работы веб-студий // XXVIII Российская научно-техническая конференция профессорско-преподавательского состава, научных сотрудников и аспирантов университета с приглашением ведущих ученых и специалистов родственных вузов и организаций – 2021. – С. 155 – 156.
2. Баклан В.И., Использование технологии CI/CD при разработке приложения «автоматизированная система коммуникаций студентов, администрации и потенциальных работодателей // Электронный сборник трудов молодых специалистов Полоцкого Государственного Университета – 2020. – С. 5 – 6.
3. Коняева О.С., Системы контроля версий в процессе работы веб-студий // XXVIII Российская научно-техническая конференция профессорско-преподавательского состава, научных сотрудников и аспирантов университета с приглашением ведущих ученых и специалистов родственных вузов и организаций – 2021. – С. 155 – 156.

4. Баклан В.И., Использование технологии CI/CD при разработке приложения «автоматизированная система коммуникаций студентов, администрации и потенциальных работодателей» // Электронный сборник трудов молодых специалистов Полоцкого Государственного Университета – 2020. – С. 5 – 6.
5. Delicheh H.O., Decan A., Mens T., A Preliminary Study of GitHub Actions Dependencies // University of Mons (UMONS), Mons, Belgium, F.R.S.-FNRS Research Associate – 2022.

References

1. Konyaeva, O.S., "Version Control Systems in the Process of Work of Web Studios", Proceedings of the XXVIII Russian Scientific and Technical Conference of Professors, Teachers, and Researchers of the University with the Invitation of Leading Scientists and Specialists of Related Universities and Organizations – 2021, pp. 155-156.
2. Baklan, V.I., "The Use of CI/CD Technology in the Development of the Application 'Automated Communication System for Students, Administration, and Potential Employers'", Electronic Collection of Works of Young Specialists of Polotsk State University – 2020, pp. 5-6.
3. Konyaeva O.S., Version control systems in the process of web studios // XXVIII Russian Scientific and technical Conference of faculty, researchers and graduate students of the University with the invitation of leading scientists and specialists of related universities and organizations - 2021. – pp. 155 – 156.
4. Baklan V.I., The use of CI/CD technology in the development of the application "automated communication system for students, administration and potential employers" // Electronic collection of works of young specialists of Polotsk State University – 2020. – pp. 5-6.
5. Delicheh, H.O., Decan, A., Mens, T., "A Preliminary Study of GitHub Actions Dependencies", University of Mons (UMONS), Mons, Belgium, F.R.S.-FNRS Research Associate – 2022.