УДК 004.457

А. Е. Федоров[1] *

Система контейнеризации

[1] Национальный исследовательский ядерный университет «МИФИ», г. Москва, Российская Федерация
* e-mail: 89032904807alek@gmail.com

Аннотация. В работе представлен программный модуль, реализующий систему контейнеризации. Рассмотрена проблематика управления безопасностью цифровых продуктов в рамках IT-инфраструктуры. Цель работы заключается в обеспечении доступности программного обеспечения с использованием технологий контейнеризации. В статье рассматривается важность внедрения контейнерных технологий. В ходе проектирования была разработана UML-диаграмма компонентов для проработки архитектуры программного модуля. Также был проведен сравнительный анализ технологий в трех областях и подобраны лучшие инструменты для разработки информационного продукта. Была реализована система контейнеризации, вмещающая в себя весь необходимый функционал для обеспечения доступности. Предложенное решение позволяет в полной мере управлять доступностью программных модулей и может быть использовано в корпоративной среде для обеспечения данного аспекта безопасности.

Ключевые слова: безопасность, виртуализация, изоляция, контейнеризация, пользовательское пространство, разработка

A. E. Fedorov[1] *

Containerization system

[1] National Research Nuclear University MEPhI, Moscow, Russian Federation
* e-mail: 89032904807alek@gmail.com

Abstract. The paper presents a software module implementing a containerization system. The issue of managing the security of digital products within the IT infrastructure is addressed. The aim of the study is to ensure the availability of software using containerization technology. The article discusses the importance of implementing container technologies. During the design process, a UML class diagram was developed to refine the architecture of the software module. Additionally, a comparative analysis of technologies in three categories was conducted, and the best tools for developing the information product were selected. A containerization system was implemented, incorporating all the necessary functionality to ensure availability. The proposed solution allows for full management of the availability of software modules and can be utilized in a corporate environment to address this aspect of security.

Keywords: security, virtualization, isolation, containerization, user space, development

Введение

В наши дни разработка программного обеспечения – сложный процесс. Цифровые продукты приобретают большой масштаб. Требования к ним вырастают. Требования к безопасности – не исключение. К сожалению, сложно обеспечить должный уровень безопасности без стороннего программного обеспечения. Это связано с масштабами цифровых продуктов.

Чтобы обеспечить нужный уровень безопасности, было разработано большое количество инструментов для каждого аспекта. Это не только облегчило задачи инженеров, но и сделало поддержку должного уровня защиты дешевле: в областях, в которых был нужен строгий контроль, исключили ручную работу; действия, требующие многократного повторения, подверглись автоматизации; для анализа информации сформировали алгоритмы [1, 2].

В данной научной статье рассмотрена компонента инфраструктуры под названием «система контейнеризации». За последние годы рынок технологии контейнеризации сильно вырос и продолжает расти. Согласно отчету “Application Container Market by End-user, Component and Geography – Forecast and Analysis 2023-2027” портала technavio.com [3], каждый год с 2017-2023г.г. размер рынка технологии в среднем растет на 20.53%. Подобный рост прогнозируется до 2027 г.

Спектр прикладных задач технологии контейнеризации обширен. Ее используют повсеместно и решают самые различные проблемы. Контейнеризация позволяет взглянуть на привычные вещи по другим углом, с новой точки зрения. Научная работа затрагивает ключевые задачи технологии: изоляция приложений, управление зависимостями, упрощение развертки приложения и т. д.

Системы контейнеризации используются в самых различных отраслях. На рис. 2 можно увидеть, как распределяется доля использования технологии контейнеризации между конечными пользователями.

Рис. 2. Круговая диаграмма распределения долей рынка между отраслями [3]

Большую часть рынка делят между собой категории «Банковское дело, финансовые услуги и страхование» и «Забота о здоровье и наука о жизни». Остальную, меньшую часть между собой делят «Телеком и информационные технологии» и «Розничная торговля и электронная коммерция».

В данном разделе мы рассмотрим проектирование программного модуля контейнеризации. Для проработки основных сущностей системы была использована UML-диаграмма компонентов.

На рис. 3 видны не только основные сущности программного модуля, но и отношения между ними. Также для каждого компонента проработаны его атрибуты и методы.

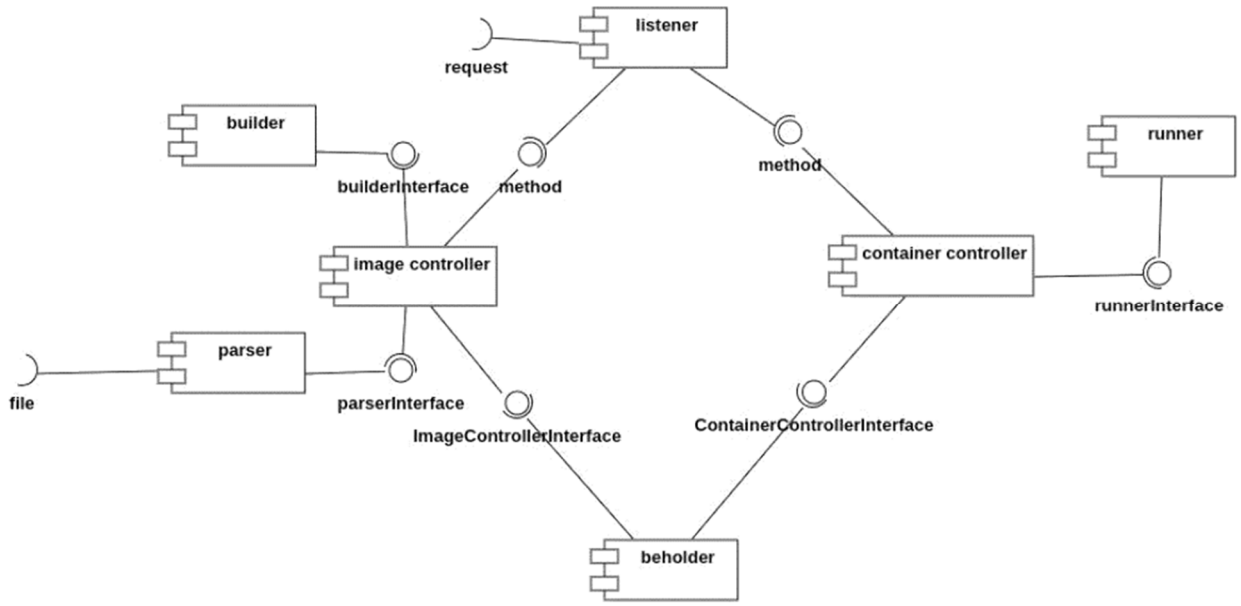

Рис. 3. Диаграмма компонентов

Для разработки модуля необходимо провести сравнительный анализ инструментов для реализации функционала. Решения должны давать возможность воспроизвести модель, построенную на рис. 3. Подобный подбор и анализ инструментов для реализации программного модуля очень важны, поскольку от этого будет зависеть возможность поддержки модуля в дальнейшем.

В рамках анализа были исследованы 3 области:

- язык программирования;
- система хранения данных;
- веб-фреймворк для взаимодействия с пользователем.

Язык программирования выбирается исходя из производительности, сложности масштабирования и соблюдения безопасности. Для веб-фреймворка важна скорость работы, безотказность и функциональность. Хранилище выбирается исходя из сложности эксплуатации и скорости доступа к данным. Эти три области выбраны для сравнения, поскольку они являются ключевыми составляющими программного модуля системы контейнеризации. От них зависит работоспособность, стабильность и безопасность программного решения. Табл. 1–3 демонстрируют сравнительный анализ в каждой из областей.

Таблица 1

Сравнение языков программирования

Язык	Производительность	Сложность масштабирования	Сложность освоения	Безопасность
C	высокая	очень сложно масштабировать [4]	средняя	большие риски из-за ручного контроля памяти
C++	ниже, чем c, но тоже высокая	проблемно масштабировать	средняя	большие риски из-за ручного контроля памяти
Python	низкая	проблемно масштабируется, если речь идет о нагрузке	простая	автоматическое управление памятью, большинство библиотек и фреймворков безопасны
Golang	на уровне C++	просто масштабируется	средняя	автоматическое управление памятью, большинство библиотек и фреймворков безопасны
Rust	на уровне C++	сложно масштабируется	высокая	на уровне C++

Таблица 2

Сравнение веб фреймворков на Go

Фреймворки	Скорость	Безотказность	Обилие функционала	Трудозатраты
gin	в 40 раз быстрее стандартной библиотеки go	предусмотрены обработчики ошибок. Сервер будет всегда доступен	есть необходимый функционал	минимальные
fiber	разработан на основе fastHTTP, поэтому может похвастаться высокой скоростью	предусмотрены обработчики ошибок. Сервер будет всегда доступен	обширный функционал	средние
Echo	не требует динамического выделения памяти [5]	предусмотрены обработчики ошибок. Сервер будет всегда доступен [5]	обширный функционал	минимальные

Таблица 3

Сравнение систем хранения данных

Система	Скорость доступа к информации	Скорость доступа к файлам	Надежность	Сложность эксплуатации	Непредсказуемость
filesystem	высокая	высокая	высокая	низкая	низкая
s3	средняя, поскольку для взаимодействия используется http [6]	средняя, поскольку для взаимодействия используется http [6]	высокая	низкая	крайне высокая, поскольку реализация s3 хранилища зависит от поставщика
data base	высокая	средняя, поскольку есть прослойка db->file	высокая	средняя	ниже среднего, поскольку система сложная, можно упустить детали

По итогам были выбраны следующие технологии:

- язык программирования – Golang [7];
- система хранения данных – файловая система;
- веб фреймворк для взаимодействия с пользователем – Echo [5].

Результаты

В ходе программной разработки была реализована система контейнеризации. Модуль включает в себя следующий функционал:

- обработка инструкций для образов;
- сборка образов по инструкции;
- создание, запуск, останов контейнера;
- механизм сетевого взаимодействия;
- поддержка разделов (volumes).

Инструкции для создания образа пишутся в yaml документе и подаются на вход системе. Модуль считывает данные и в соответствии с ними конструирует образ контейнера. Образ строится на базе файловой системы overlayFS [8, 9] версии 2 и состоит из «слоев». Каждый слой может быть одного из 7 типов [10]. На выходе получается GZip архив [11], который содержит в себе всю необходимую информацию для запуска контейнера.

Для запуска контейнера необходимо иметь готовый образ. Запуск происходит следующим образом:

- отделяется пользовательское пространство процессов [9] с помощью утилиты unshare [12];
- меняется корневой каталог с помощью встроенного механизма Linux – chroot [12];

- создаются cgroups для ограничения ресурсов контейнера [13];
- запускается опорный процесс.

Дальнейшее управление контейнеров производится с помощью запуска и останова опорного процесса.

Сетевое взаимодействие контейнера с хостовой машиной осуществляется посредством создания двух сетевых интерфейсов [14, 15]. Первый присоединяется в контейнер, второй – на хостовую машину. Связка портов осуществляется с помощью инструмента iptables. Работа с разделами происходит посредством «бинда» внешней директории на внутреннюю с помощью команды Linux – mount.

Также разработан интерфейс для взаимодействия с пользователем. Он представляет собой веб-сервер, реализующий методологию REST API на базе протокола HTTP. Данная технология стандартизирует виды запросов, что помогает разделять ответственность между методами.

Обсуждение и заключение

В данной работе проводилось исследование и подготовка к программной реализации системы контейнеризации. Было установлено определение системы контейнеризации и объяснена её целесообразность. Выделены области компетенции, которые охватывает данная система. Был произведен обзор актуальности данной технологии.

Проведено моделирование системы контейнеризации. Составлена диаграмма компонентов. С ее помощью получилось составить более полную архитектурную картину работы модуля. Для каждой сущности были составлены компоненты. Более подробное взаимодействие сущностей было проработано отдельно. В результате выявлены зоны ответственности и роли каждой части системы.

Кроме того, был произведен сравнительный анализ технологий, необходимых для реализации системы контейнеризации. Выделены три категории, в каждой из которых проводился сравнительный анализ. В результате были подобраны лучшие инструменты для реализации программного модуля. Помимо этого, были проработаны прикладные задачи для каждой из технологий.

В качестве результата был разработан программный модуль системы контейнеризации, включающий в себя сборку образов, создание и управление контейнерами, а также подключение сетевых интерфейсов и разделов. Вдобавок описан интерфейс взаимодействия пользователя с системой.

Таким образом, разработка системы контейнеризации является важным шагом в обеспечении доступности программных модулей в современном мире информационных технологий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Боротько Т. В. // XVII международная научно-практическая конференция "управление информационными ресурсами". – 2021. – С.208-209.

2. Басыня, Е. А. Автоматизированная установка и конфигурирование серверных решений / Е. А. Басыня, М. С. Лукина // Современные материалы, техника и технологии. – 2016. – No 2(5). – С. 21-26.

3. Application Container Market by End-user, Component and Geography- Forecast and Analysis 2023-2027 // technavio. – URL: https://www.technavio.com/report/application-container-market-analysis

4. Документация по языку C // cppreference. – URL: https://devdocs.io/c/

5. Документация по фреймворку Echo // Официальный сайт LabStack LLC. – URL: https://echo.labstack.com/docs

6. Документация по технологии S3 // Официальный сайт Amazon Web Services, Inc. – URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

7. Golang documentation // Официальный сайт Google Inc. – URL: https://go.dev/doc/

8. Документация по файловой системе OverlayFS // kernel.org. – URL: https://docs.kernel.org/filesystems/overlayfs.html

9. Исходный код ядра Linux // Elixir. – URL: https://elixir.bootlin.com/linux/latest/source

10. Забавский, В. В. Уязвимости в технологии контейнеризации docker / В. В. Забавский, Т. В. Борботько // Управление информационным ресурсами : Материалы XVII Международной научно-практической конференции, Минск, 12 марта 2021 года. – Минск: Академия управления при Президенте Республики Беларусь, 2021. – С. 208-209. – EDN ZBUNUL.

11. Кришна М. Кумар. Обеспечение безопасности контейнеров во время выполнения с использованием предложенной модели карты управления безопасностью // Глобальная конференция по развитию технологий (ГКАТ). – Бангалор, Индия, 2019. – С. 1-6.

12. Горбачева Т. Ф. Контейнеризация и докер-платформа в инновационных технологиях / Горбачева Т. Ф. // Информационно-телекоммуникационные системы и технологии (ИТСИТ-2018). - Кемерово, 2018. - С. 189-190.

13. Лиз Райс. Безопасность контейнеров. Фундаментальный подход к защите контейнеризированных приложений: учебное пособие. – 2021. – 224 с.

14. Басыня Е. А. Вопросы управления трафиком в оверлейных сетях // Автоматика и программная инженерия. – 2014. – №. 3 (9). – С. 29-32.

15. Басыня Е.А. Сетевая информационная безопасность: Учебник. – Москва : НИЯУ МИФИ, 2023. – 72 с.