Программные системы и вычислительные методы
*Правильная ссылка на статью:*
Шейнман В., Стариков Д.Д., Тюменцев Д.В., Вавилов Г.Д. Повышение эффективности процессов разработки программного обеспечения: контейнерные технологии // Программные системы и вычислительные методы. 2024. № 4. DOI: 10.7256/2454-0714.2024.4.72015 EDN: JXRTJC URL: https://nbpublish.com/library_read_article.php?id=72015

# Повышение эффективности процессов разработки программного обеспечения: контейнерные технологии

**Шейнман Вред**

ORCID: 0009-0003-5977-4211

независимый исследователь

3498838, Израиль, г. Хайфа, ул. Абба Хоши, 199

vered.sheinman@gmail.com

**Стариков Дмитрий Дмитриевич**

ORCID: 0009-0009-8788-709X

студент; институт № 8 «Компьютерные науки и прикладная математика»; Московский авиационный институт (национальный исследовательский университет)

125993, Россия, г. Москва, Волоколамское шоссе, 4

dd.starikov@gmail.com

**Тюменцев Денис Викторович**

ORCID: 0009-0003-5275-3223

независимый исследователь

670013, Россия, республика Бурятия, г. Улан-Удэ, ул. Ключевская, 40В

tyumencev_dv@rambler.ru

**Вавилов Георгий Давидович**

ORCID: 0000-0002-5951-3201

студент; отделение радиотерапии; Национальный медицинский исследовательский центр имени академика Е.Н. Мешалкина

630055, Россия, г. Новосибирск, ул. Речкуновская, 15

Frost20@narod.ru

Статья из рубрики "Показатели качества и повышение надежности программных систем"

**DOI:**

10.7256/2454-0714.2024.4.72015

**EDN:**

JXRTJC

**Дата направления статьи в редакцию:**

17-10-2024

**Аннотация:** В статье авторы рассматривают влияние контейнерных технологий на процессы разработки программного обеспечения (ПО). Основное внимание уделяется роли контейнеризации в оптимизации процессов развертывания и управления приложениями, а также в повышении гибкости и масштабируемости программных систем. В исследовании анализируются ключевые аспекты применения контейнеров, включая изоляцию приложений, повышение переносимости ПО между различными средами, а также снижение затрат на эксплуатацию благодаря оптимизации использования вычислительных ресурсов. Рассматриваются такие современные инструменты, как Docker и Kubernetes, которые позволяют стандартизировать и автоматизировать процессы развертывания и управления инфраструктурой. Аторы обсуждают примеры практического применения контейнерных технологий в крупных российских и иностранных компаниях, где контейнеризация значительно улучшила процессы разработки и эксплуатации ПО. Для анализа эффективности контейнерных технологий использованы методы сравнительного анализа, позволяющие оценить их влияние на гибкость инфраструктуры и производительность программных систем. Источниками данных служили научные публикации. Новизна исследования заключается в рассмотрении применения контейнерных технологий в контексте современных практик разработки ПО, что позволяет значительно ускорить процессы разработки, тестирования и развертывания программных продуктов. Полученные результаты показывают, что контейнеризация способствует улучшению производительности систем, упрощает управление приложениями и снижает затраты на эксплуатацию. Примеры практического использования Docker и Kubernetes в крупных компаниях демонстрируют, что контейнеризация значительно повышает гибкость инфраструктуры и масштабируемость решений, позволяя разработчикам легко адаптироваться к меняющимся условиям и требованиям рынка. В заключение подчеркивается, что контейнерные технологии играют ключевую роль в современных процессах разработки ПО, и их дальнейшее развитие будет способствовать еще более значительным улучшениям в области автоматизации и управления инфраструктурой программных систем.

**Ключевые слова:**

контейнеризация, разработка программного обеспечения, Docker, Kubernetes, масштабируемость, оптимизация ресурсов, эксплуатация программного обеспечения, автоматизация, платформы, изоляция процессов

## Введение

Контейнерные технологии играют важную роль в разработке программного обеспечения (ПО), предоставляя эффективные решения для управления приложениями и инфраструктурой. В условиях роста сложности программных систем и стремительного темпа изменений в ИТ-индустрии контейнеризация обеспечивает необходимые инструменты для оптимизации процессов создания, тестирования и развертывания

элементов ПО.

Контейнерные платформы, такие как Docker, а также системы оркестрации контейнеров, включая Kubernetes, стали стандартом в области разработки. Эти технологии позволяют изолировать приложения и их зависимости в независимые, легковесные среды, что способствует повышению гибкости и переносимости ПО между различными вычислительными средами. Это устраняет проблемы, связанные с несовместимостью конфигураций и инфраструктуры, и упрощает управление жизненным циклом программных продуктов. Цель данной статьи – проанализировать, каким образом использование контейнерных технологий способствует повышению эффективности процессов разработки ПО.

## Основная часть. Понятие и особенности контейнеризации

Методология создания и развертывания программных приложений в изолированных средах называется контейнеризацией. Контейнеры предоставляют средства для упаковки ПО и всех его зависимостей в самодостаточные, изолированные от хост-системы единицы, что позволяет значительно упростить переносимость и воспроизводимость приложений между различными вычислительными платформами [1]. В отличие от традиционной виртуализации, контейнеры используют общее ядро операционной системы, обеспечивая при этом полную изоляцию процессов, файловой системы и сетевых ресурсов для каждого элемента.

Контейнеры используют механизмы изоляции, такие как cgroups и namespaces, которые позволяют ограничить доступ к ресурсам и создать изолированные пространства имен для процессов, сетевых интерфейсов и систем [2]. Это гарантирует, что файлы, работающие в контейнерах, не будут влиять друг на друга, обеспечивая высокий уровень безопасности и изоляции без необходимости в полноценной виртуализации.

Контейнеризация предоставляет множество преимуществ для разработки и эксплуатации ПО. Это различные аспекты работы с приложениями – от повышения гибкости до оптимизации использования ресурсов (таблица 1).

**Таблица 1. Преимущества контейнеризации [3, 4]**

| Преимущество | Описание | Применение на практике |
|---|---|---|
| Переносимость | Контейнеры обеспечивают возможность запуска продуктов на различных платформах без изменения кода. | Программисты могут разрабатывать ПО локально и затем без изменений развертывать его в облаке. |
| Масштабируемость | Легкость горизонтального масштабирования приложений путем добавления новых контейнеров. | Использование контейнеров для динамического увеличения числа экземпляров приложения при высокой нагрузке. |
| Изоляция | Полная изоляция процессов, файловых систем и сетевых ресурсов, что предотвращает конфликты. | Одновременный запуск нескольких файлов на одном сервере без риска конфликтов или помех. |
| Эффективное | Контейнеры используют | Повышение плотности |

| | | |
|---|---|---|
| использование ресурсов | общее ядро операционной системы, снижая затраты на ресурсы по сравнению с виртуальными машинами. | развертывания на физических серверах и снижение затрат на инфраструктуру. |
| Гибкость | Упрощение разработки и тестирования благодаря возможности быстро создавать и изменять контейнеры. | Быстрое тестирование и развертывание новых функций без прерывания основной работы продукта. |

Преимущества контейнеризации, представленные в таблице, подчеркивают ее важность для современного процесса разработки и эксплуатации ПО. Более эффективное использование ресурсов и быстрота развертывания способствуют оптимизации инфраструктуры и уменьшению затрат на поддержание приложений. В совокупности, это делает контейнеризацию мощным инструментом для ускорения инноваций и повышения общей эффективности в ИТ-индустрии.

**Основные технологии и инструменты контейнеризации**

Согласно онлайн-опросу, проведенному американской компанией Statista в 2024 году, одной из самых популярных и востребованных технологий для контейнеризации является **Docker** – о его использовании сообщили 59% разработчиков [5]. Он предоставляет средства для создания, управления и развертывания контейнеров, позволяя легко упаковывать программные продукты вместе с их зависимостями в стандартные контейнерные образы (рис.1).

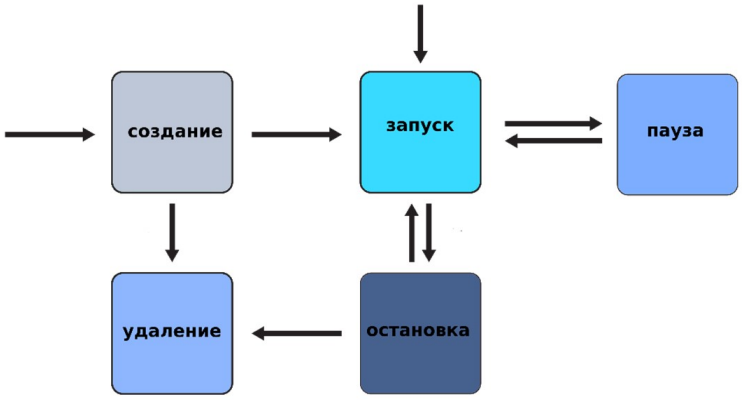


Рисунок 1. Основные состояния контейнеров в Docker

Компоненты Docker включают Docker Engine, Docker Hub и Docker Compose [6]. **Docker Engine** является ключевым элементом платформы, отвечающим за создание и выполнение контейнеров и взаимодействие с операционной системой. **Docker Hub** представляет собой облачную платформу для хранения и распространения контейнерных образов. Она позволяет разработчикам делиться готовыми элементами ПО с другими пользователями, а также загружать необходимые продукты из общего хранилища. **Docker Compose** является инструментом, который упрощает работу с многоконтейнерными приложениями, предоставляя возможность описывать конфигурации в виде YAML-файлов и управлять процессами их запуска и взаимодействия.

Встроенная в Docker система оркестрации контейнеровназывается **Docker Swarm** [7].

Она обеспечивает простую интеграцию с существующими Docker-средами и позволяет распределять контейнеры между несколькими узлами, поддерживая балансировку нагрузки и автоматическое восстановление.

**Kubernetes** – это система оркестрации контейнеров с открытым исходным кодом, предназначенная для автоматизации процессов развертывания, управления и масштабирования программных продуктов, построенных на контейнерах. Она разработана для эффективной работы с крупными распределенными системами и предлагает мощные инструменты для управления большим числом контейнеров в различных вычислительных окружениях [8]. Это делает Kubernetes незаменимым инструментом для компаний, стремящихся к высокому уровню автоматизации и гибкости в разработке ПО (рис.2).

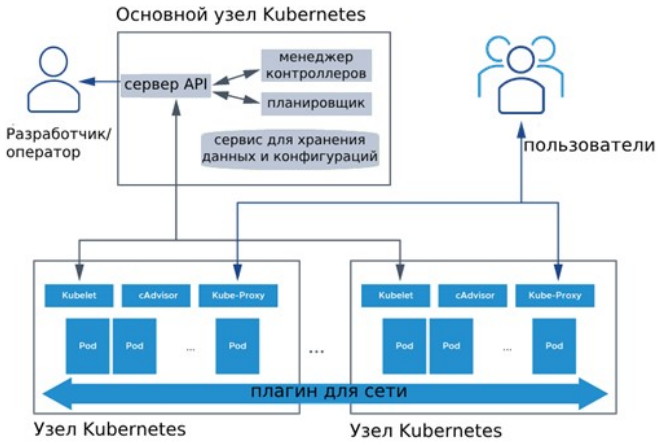


Рисунок 2. Архитектура системы Kubernetes

Разработчики и операторы взаимодействуют с системой через API сервер для управления контейнерными приложениями. Рабочие узлы содержат поды (Pod), которые представляют собой группы контейнеров, а также компоненты для мониторинга и управления – Kubelet, cAdvisor и Kube-Proxy. Узлы связаны через сетевые плагины, обеспечивающие взаимодействие контейнеров в кластере.

Согласно исследованиям Statista, в 2023 году 38% разработчиков отмечали повышение производительности, доступности и отказоустойчивости систем как главное преимущество Kubernetes. Ключевые возможности технологии включают автоматическое масштабирование, оркестрацию сетей и хранилищ, автовосстановление и балансировку нагрузки [9]. **Автоматическое масштабирование** позволяет системе динамически изменять количество контейнеров в зависимости от текущей нагрузки на приложение, что способствует эффективному использованию ресурсов. **Оркестрация сетей и хранилищ** обеспечивает управляемое взаимодействие между контейнерами и предоставляет доступ к внешним хранилищам данных, упрощая работу с распределенными файлами. **Функция автовосстановления** автоматически перезапускает или заменяет контейнеры в случае их выхода из строя, поддерживая бесперебойную работу системы. **Балансировка нагрузки** распределяет входящий трафик между контейнерами, что позволяет достичь оптимальной производительности продуктов, даже при высокой нагрузке.

**OpenShift** –это платформа контейнеризации, основанная на Kubernetes. Она предоставляет дополнительные инструменты для управления безопасностью, разработкой и эксплуатацией файлов в контейнерах (рис.3).

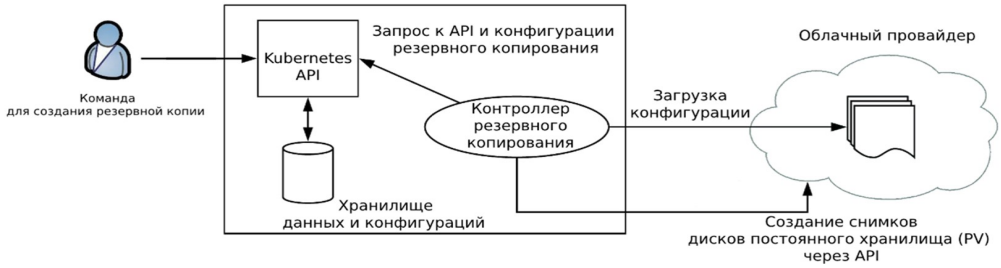


Рисунок 3. Схема OpenShift

OpenShift ориентирован на корпоративный сегмент и включает средства для разработки приложений [10]. Пользователь инициирует резервное копирование через команду, которая взаимодействует с Kubernetes API. Контроллер резервного копирования управляет процессом, сохраняя конфигурации в хранилище и загружая данные в облачный провайдер с созданием снимков дисков. Технология предлагает улучшенные механизмы управления безопасностью контейнеров и процессами аутентификации, а также интеграцию с CI/CD (непрерывной интеграцией и развертыванием), что делает ее эффективным инструментом для крупных организаций с комплексными требованиями к инфраструктуре.

**CRI-O** – это проект с открытым исходным кодом, специально разработанный для использования с Kubernetes. Он предоставляет минималистичную, но эффективную среду выполнения контейнеров, что позволяет Kubernetes напрямую управлять контейнерами без необходимости использования более сложных решений, таких как Docker (рис.4)

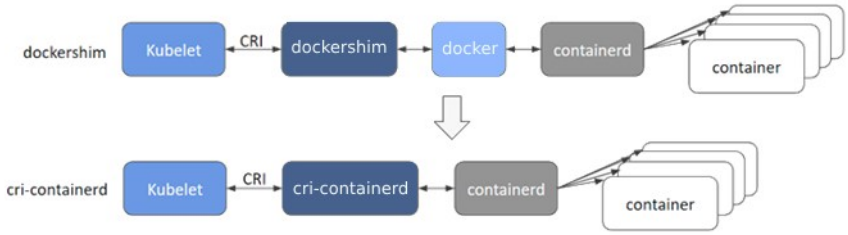


Рисунок 4. Сравнение взаимодействия Kubernetes с контейнерами через dockershim и cri-containerd

На схеме показаны два различных подхода взаимодействия Kubernetes с контейнерами. В первом случае используется dockershim – промежуточный слой, который позволяет Kubernetes взаимодействовать с контейнерами через Docker. В этой архитектуре Kubernetes с помощью компонента Kubelet отправляет команды через интерфейс CRI (Container Runtime Interface) к Docker, который управляет контейнерами через низкоуровневый компонент containerd. Контейнеры создаются и управляются Docker, что добавляет лишний уровень абстракции.

Во втором подходе используется CRI-O. В этом случае Kubelet взаимодействует напрямую с CRI-O через тот же интерфейс CRI, что исключает необходимость использования Docker. Вместо этого CRI-O напрямую управляет контейнерами через containerd, что упрощает архитектуру, улучшает производительность и устраняет избыточные слои. CRI-O сосредоточен на минимизации функциональности для выполнения только тех задач, которые необходимы для работы с Kubernetes. Это делает его более легковесной и эффективной альтернативой для Kubernetes-кластеров [11].

**Podman** – это инструмент с открытым исходным кодом, обладающий функциональностью, схожей с Docker, но с некоторым отличием: Podman не требует наличия фонового

процесса для управления контейнерами. Это делает его более безопасной альтернативой Docker, особенно в средах с повышенными требованиями к безопасности (рис.5)

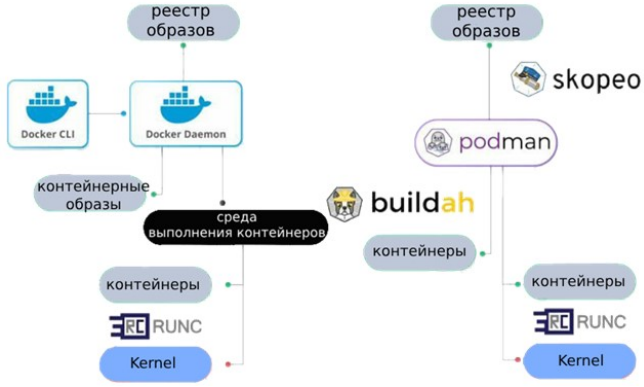


Рисунок 5. Сравнение архитектуры Docker и Podman

Podman интегрируется с другими инструментами, такими как Buildah для создания контейнерных образов и Skopeo для их управления, что делает его надежным инструментом для работы с контейнерами в средах с высокими требованиями к безопасности.

Согласно исследованию, проведенному в Национальном научно-вычислительном центре энергетических исследований (NERSC) в сотрудничестве с американским производителем ПО Red Hat, технология Podman адаптирована для использования в области высокопроизводительных вычислений (HPC). Podman предлагает такие преимущества, как возможность запуска контейнеров без привилегий root, что повышает уровень безопасности, а также возможность сборки контейнерных образов непосредственно на узлах суперкомпьютера Perlmutter [12]. Результаты тестирования различных режимов работы Podman, включая podman-exec и Podman container-per-process, продемонстрировали производительность, сопоставимую с непосредственным использованием аппаратных ресурсов, что делает Podman перспективным инструментом для использования.

**Применение контейнеров на различных этапах разработки ПО**

Контейнеризация предоставляет разработчикам возможность эффективно управлять программными системами на всех стадиях их жизненного цикла – от разработки до эксплуатации. Контейнеры помогают создавать изолированные среды, которые можно быстро развертывать и тестировать, что особенно важно для крупных компаний [13].

**На этапе разработки** контейнеры используются для создания стандартизированных сред, которые обеспечивают одинаковую работу приложения на различных устройствах. Например, американская компания **Google** использует контейнеризацию для разработки своих сервисов, включая Google Search и Gmail [14]. Контейнеры позволяют разработчикам создавать приложения в идентичных продуктивным средах, что исключает проблемы, связанные с различиями в конфигурации операционных систем и библиотек. Это существенно упрощает процесс написания кода и его последующее тестирование. Google Cloud предоставляет экосистему, необходимую для более быстрой разработки и внедрения ПО без ущерба для безопасности. Согласно отчетам, 98 % клиентов компаний, которые используют инструменты Google для контейнеризации, развертывают приложение с первой попытки менее чем за 5 минут.

**Тестирование ПО** – это важнейший этап разработки, на котором контейнерные технологии позволяют создавать изолированные и воспроизводимые тестовые среды. В американском стриминговом сервисе **Netflix** контейнеры играют ключевую роль в ускорении тестирования и улучшении качества новых версий приложений и сервисов. Для этих целей Netflix использует Docker и Kubernetes [15]. С помощью Docker компания создает контейнеры, в которых изолируются различные версии приложений, что позволяет проводить тестирование в идентичных средах, избегая проблем, связанных с различиями в конфигурации. Kubernetes помогает Netflix масштабировать тестовые среды для имитации различных сценариев работы потокового сервиса под разными уровнями нагрузки. Это позволяет тестировать производительность и устойчивость приложений к высоким нагрузкам, быстрее выявлять потенциальные ошибки и снижать риски, связанные с обновлениями приложений.

Российская компания **«Яндекс»** применяет контейнеризацию на этапе тестирования для проверки новых функциональных возможностей своих продуктов. Для этих целей Яндекс использует такие инструменты, как Docker и Kubernetes. Docker позволяет создавать изолированные контейнеры, в которых разрабатываются и тестируются отдельные компоненты сервисов [16]. Эти контейнеры помогают воспроизводить тестовые среды, идентичные реальным, что снижает вероятность возникновения ошибок.

Kubernetes используется для автоматизации развертывания контейнеров и управления ими на кластерах серверов. Это позволяет Яндексу масштабировать тестовые окружения и проводить нагрузочные тесты для проверки производительности новых функций под разными уровнями нагрузки. Благодаря такой интеграции Docker и Kubernetes, компания может параллельно тестировать множество версий своих продуктов, что ускоряет цикл разработки и снижает риски внедрения обновлений в рабочие среды.

На этапе **развертывания и эксплуатации** ПО контейнеризация предоставляет значительные преимущества, обеспечивая гибкость и масштабируемость приложений. Примером этого может служить практика **American Express (США)** которая активно использует контейнеризацию в процессах разработки и эксплуатации ПО [17]. Компания применяет Docker и Kubernetes для автоматизации развертывания финансовых сервисов, что позволяет им гибко масштабировать приложения в зависимости от изменяющихся требований и нагрузки на систему.

После развертывания контейнеры продолжают играть важную роль в **эксплуатации и масштабировании ПО,** обеспечивая автоматическое управление ресурсами и гибкость в изменении конфигураций.

## Заключение

Контейнерные технологии способствуют повышению эффективности разработки ПО за счет изоляции приложений и их зависимостей, что облегчает переносимость между различными средами. Это позволяет устранить проблемы совместимости, ускоряя процессы разработки, тестирования и развертывания. Использование контейнеров оптимизирует распределение ресурсов, снижая нагрузку на инфраструктуру и улучшая масштабируемость программных продуктов. Такие платформы, как Docker и Kubernetes, обеспечивают автоматизацию и гибкость управления файлами, что сокращает время на внедрение новых функций и минимизирует затраты на поддержание инфраструктуры.

**Финансирование.** Исследование не имело спонсорской поддержки.

**Вклад авторов.** Все авторы внесли равный вклад в написание настоящей статьи.

**Конфликт интересов.** Авторы заявляют об отсутствии конфликта интересов.

**Funding.** The study had no sponsorship.

**Authors contributions.** All authors contributed equally to this article.

**Conflict of interest.** The authors declare no conflict of interest.

## Библиография

1. Белодед Н.И., Демиденко К.Г. Развитие и применение технологии контейнеризации в разработке программного обеспечения // Актуальные проблемы научных исследований: теоретические. – 2023. – С. 57.
2. Bondarenko A.S., Zaytsev K.S. Using container management systems to build distributed cloud information systems with microservice architecture // International Journal of Open Information Technologies. – 2023. – V. 11. – № 8. – P. 17-23.
3. Можаровский Е.А. Разработка мобильных приложений: от идеи до рынка // Современные научные исследования и инновации. – 2024. – № 1.
4. Aluev A. Scalable web applications: a cost-effectiveness study using microservice architecture // Cold Science. – 2024. – № 8. – С. 32-38.
5. Muzumdar P, Bhosale A., Basyal G., Kurian G. Navigating the Docker ecosystem: a comprehensive taxonomy and survey // arXiv preprint arXiv:2403.17940. – 2024.
6. Christudas B.A. Introducing Docker // Java Microservices and containers in the Cloud: with Spring Boot, Kafka, PostgreSQL, Kubernetes, Helm, Terraform and AWS EKS. – Berkeley, CA: Apress, 2024. – P. 281-343.
7. Higgins T., Jha D.N., Ranjan R. Swarm Storm: an automated chaos tool for docker swarm applications // Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing. – 2024. – P. 367-369.
8. Миронов Т.О. Построение информационной архитектуры системы автоматизации цикла выпуска программного обеспечения // Инжиниринг предприятий и управление знаниями. – С. 280.
9. Дудак А.А. Сравнительный анализ инструментов разработки для систем управления проектами: преимущества стек-технологий TypeScript и React // Новая наука: от идеи к результату. – 2024. – № 9. – С. 32-40.
10. Глумов К.С. Шаблоны, лежащие в основе java, kubernetes и современных распределенных систем // Хлебопечение России. – 2024. – Т. 68. – № 1. – С. 6-12.
11. Poggiani L., Puliafito C., Virdis A., Mingozzi E. Live Migration of Multi-Container Kubernetes Pods in Multi-Cluster Serverless Edge Systems // Proceedings of the 1st Workshop on Serverless at the Edge. – 2024. – P. 9-16.
12. Stephey L., Canon S., Gaur A., Fulton D., Younge A. Scaling Podman on Perlmutter: Embracing a community-supported container ecosystem // 2022 IEEE/ACM 4th International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC). – IEEE, 2022. – P. 25-35.
13. Sidorov D. Leveraging web components for scalable and maintainable development // Sciences of Europe. – 2024. – № 150. – P. 87-89.
14. Чередников К. А., Лаврова Е. Д., Марухленко А. Л. Современный взгляд на контейнеризацию // Современные информационные технологии и информационная безопасность. – 2023. – P. 115-119.
15. Erdenebat B., Bud B., Kozsik T. Challenges in service discovery for microservices deployed in a Kubernetes cluster – a case study // Infocommunications Journal. – 2023. – V. 15. – № SI. – P. 69-75.

16. Макарова Н.В., Савичев Д.Е. Применение методов искусственного интеллекта при эксплуатации программного обеспечения // Актуальные проблемы экономики и управления. – 2023. – № 1. – С. 17.
17. Косарев В.Е., Добридник С.Л. Практические аспекты разработки и внедрения цифрового рубля в банковские информационные системы // Инновации и инвестиции. – 2023. – № 2. – С. 143-149.

## Результаты процедуры рецензирования статьи

*В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.*
*Со списком рецензентов издательства можно ознакомиться здесь.*

Рецензируемая работа посвящена Повышению эффективности процессов разработки программного обеспечения за счет применения контейнерных технологий.
Методология исследования базируется на изучении и обобщении научных публикаций по рассматриваемой теме и практики применения различных инструментов контейнеризации.
Актуальность работы обусловлена увеличением количества разрабатываемых компьютерных программ и ростом их сложности, что требует повышения эффективности процессов разработки программного обеспечения, которое может быть обеспечено в частности за счет применения контейнерные технологий в разработке программного обеспечения.
Научная новизна рецензируемого исследования состоит в выводах о том, что контейнерные технологии способствуют повышению эффективности разработки ПО за счет изоляции приложений и их зависимостей, что облегчает переносимость между различными средами, это позволяет устранить проблемы совместимости, ускоряя процессы разработки, тестирования и развертывания, оптимизирует распределение ресурсов, снижая нагрузку на инфраструктуру и улучшая масштабируемость программных продуктов.
В тексте статьи выделены следующие разделы: Введение, Основная часть. Понятие и особенности контейнеризации, Основные технологии и инструменты контейнеризации, Применение контейнеров на различных этапах разработки ПО, Заключение и Библиография.
В статье под контейнеризацией понимается методология создания и развертывания программных приложений в изолированных средах; отмечено, что контейнеры предоставляют средства для упаковки программного обеспечнения и всех его зависимостей в самодостаточные, изолированные от хост-системы единицы, что позволяет значительно упростить переносимость и воспроизводимость приложений между различными вычислительными платформами; в отличие от традиционной виртуализации, контейнеры используют общее ядро операционной системы, обеспечивая при этом полную изоляцию процессов, файловой системы и сетевых ресурсов для каждого элемента. В публикации детально рассмотрены преимущества контейнеризации: переносимость, масштабируемость, изоляция, эффективное использование ресурсов, гибкость. Авторами освещены особенности применения таких инструментов контейнеризации как Docker со встроенной системой оркестрации контейнеров Docker Swarm, а также Kubernetes – системы оркестрации контейнеров с открытым исходным кодом, предназначенной для автоматизации процессов развертывания, управления и масштабирования программных продуктов, построенных на контейнерах, а также платформы контейнеризации OpenShift и CRI-O. Также в статье отражены возможности Контейнеризации для эффективного управления программными

системами на всех стадиях их жизненного цикла – от разработки до эксплуатации.

Библиографический список включает 17 источников – научные публикации на русском и английском языках, на которые в тексте приведены адресные отсылки, что подтверждает наличие апелляции к оппонентам.

Рецензируемый материал соответствует направлению журнала «Программные системы и вычислительные методы», отражает результаты проведенной авторами работы, может вызвать интерес у читателей, поскольку содержит интересные сведения о применении контейнерных технологий в разработке программного обеспечения, а поэтому рекомендуется к опубликованию.