ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.42

ВЛИЯНИЕ СРЕДЫ ВИРТУАЛИЗАЦИИ DOCKER НА СОВРЕМЕННЫЕ IT-ИНФРАСТРУКТУРЫ

Москаленко Павел Анатольевич,
Магистрант 519 группы Санкт-Петербургского государственного университета промышленных технологий и дизайна. Высшая школа технологии и энергетики, Санкт-Петербург, ул. Ивана Черных, 4
E-mail: pavellange03@mail.ru

Нашук Дмитрий Сергеевич,
Студент 432 группы Санкт-Петербургского государственного университета промышленных технологий и дизайна. Высшая школа технологии и энергетики, Санкт-Петербург, ул. Ивана Черных, 4
E-mail: dnashuk@mail.ru

Аннотация

В данной статье представлен обзор эволюции и влияния среды виртуализации Docker на современные IT-инфраструктуры, начиная с ее основных компонентов и заканчивая интеграцией с облачными платформами. Описываются ключевые инструменты и технологии, такие как Docker Compose, Docker Swarm и Kubernetes, а также их роль в управлении и оркестрации контейнеров. Особое внимание уделяется вопросам безопасности контейнерных технологий и методам их повышения. В заключительной части статьи рассматриваются текущие тенденции и перспективы развития контейнерных технологий, включая серверлесс-архитектуры и мультикластерные решения.

Ключевые слова: Docker, виртуализация, IT-инфраструктуры, облачные платформы, оркестрация, Kubernetes, контейнерные технологии

IMPACT OF DOCKER VIRTUALISATION ENVIRONMENT ON MODERN IT-INFRASTRUCTURES

Pavel A. Moskalenko,
Master's student of group 519,
St. Petersburg State University of Industrial Technology and Design.
Higher School of Technology and Energy, St. Petersburg, Ivan Chernykh Street, 4.
E-mail: pavellange03@mail.ru

Dmitry S. Nashuk,
Student of group 432,
St. Petersburg State University of Industrial Technology and Design.
Higher School of Technology and Energy, St. Petersburg, Ivan Chernykh Street, 4.

E- mail: dnashuk@mail.ru

ABSTRACT

This article provides an overview of the evolution and impact of Docker virtualization on modern IT infrastructures, starting from its core components to its integration with cloud platforms. It describes key tools and technologies such as Docker Compose, Docker Swarm, and Kubernetes, and their roles in container management and orchestration. Special attention is given to container security issues and methods to enhance it. The final part of the article discusses current trends and future prospects of container technologies, including serverless architectures and multi-cluster solutions.

Keywords: Docker, virtualization, IT infrastructures, cloud platforms, orchestration, Kubernetes, container technologies

Виртуализация зародилась в 1960-х годах с системами разделения времени от IBM, позволившими многим пользователям одновременно работать на одном компьютере. В 1990-х гипервизоры, такие как VMware, сделали возможным запуск нескольких операционных систем на одном сервере, что повысило гибкость и эффективность управления ИТ-инфраструктурой.

Виртуальные машины, несмотря на свои преимущества, требовали значительных ресурсов. Контейнеризация, появившаяся в начале 2000-х, решила эту проблему, позволяя запускать приложения в изолированных средах без эмуляции полноценной ОС. Первые шаги включали chroot в Unix, FreeBSD Jails и Solaris Zones, которые обеспечивали изоляцию, но были сложны в настройке и использовании.

В 2013 году технология Docker совершила настоящий прорыв в области контейнеризации, сделав её значительно более доступной и удобной в использовании [1]. Эта платформа кардинально упростила процессы создания, деплоя и управления контейнерами, предложив инновационный подход к упаковке приложений вместе с их зависимостями в единые, легко транспортируемые модули. Централизованный репозиторий Docker Hub существенно ускорил цикл разработки и внедрения, предоставив возможность использовать готовые решения и адаптировать их под конкретные задачи.

Одним из ключевых преимуществ Docker стала его компактность: контейнеры работали на основе общей операционной системы, что существенно снижало нагрузку на системные ресурсы и позволяло размещать большее количество приложений на одном сервере. Появление этой технологии стало важной вехой в развитии контейнерных решений, оптимизировав процесс разработки, тестирования и развёртывания программного обеспечения, а также создало основу для последующего появления мощных систем оркестрации, таких как Kubernetes.

Docker Engine представляет собой клиент-серверное приложение, состоящее из трёх основных элементов: демона Docker (Docker Daemon), командной строки интерфейса (Docker CLI) и программного интерфейса Docker API (см. рис. 1).

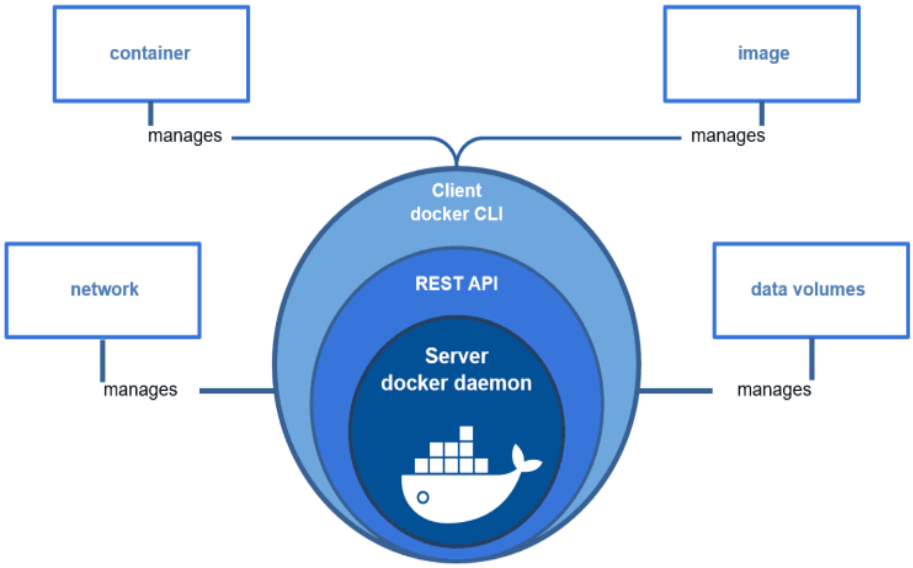

Рисунок 1. Архитектура Docker Engine

Docker Daemon (dockerd) отвечает за выполнение ключевых задач по формированию, запуску и контролю контейнеров. Командная строка Docker (Command Line Interface, CLI) обеспечивает взаимодействие пользователя с демоном Docker через терминал, позволяя отправлять команды и получать соответствующие результаты их выполнения. Программный интерфейс Docker API предоставляет возможность интеграции Docker в сторонние приложения и системы, создавая дополнительные каналы для обмена данными между компонентами [2].

Хотя Docker-контейнеры и виртуальные машины (VM) выполняют схожую функцию изоляции программного обеспечения, их архитектурные особенности и подходы к использованию системных ресурсов значительно различаются. Эти отличия подробно описаны в таблице 1.

Такое разделение ролей между компонентами Docker Engine позволяет создать эффективную систему управления контейнерами, где каждый элемент отвечает за свою специализированную задачу, а различия между контейнеризацией и виртуализацией подчеркивают уникальные преимущества каждой технологии.

Таблица 1 – Сравнительный анализ Docker контейнеров и виртуальных машин.

Характеристика	Docker контейнеры	Виртуальные машины
Изоляция	На уровне процессов	Полная виртуализация ОС
Потребление ресурсов	Низкое	Высокое
Время запуска	Миллисекунды	Минуты
Образ	Легковесный, на уровне мегабайтов	Тяжелый, на уровне гигабайтов
Совместимость	Зависит от хостовой ОС	Независима от хостовой ОС

Виртуальные машины работают на гипервизоре и включают в себя полную операционную систему, что требует значительных ресурсов [3]. Контейнеры, напротив, используют общее ядро хостовой ОС и изолируют приложения на уровне процессов, что делает их более легковесными и эффективными.

В механике работы технологии определили ряд основных компонентов и обозначили их взаимодействие:

Центральным элементом всей системы является Docker Daemon (dockerd), который осуществляет управление контейнерами, образами, сетевыми настройками и томами. Этот процесс отвечает за обработку запросов, поступающих через интерфейс Docker API. Для взаимодействия с пользователем используется Docker CLI – интерфейс командной строки, через который отправляются и исполняются команды, такие как «docker run», «docker build» или «docker pull». Программный интерфейс Docker API позволяет интегрировать функционал Docker в сторонние приложения, предоставляя доступ ко всем возможностям, которые доступны через CLI. Docker Hub представляет собой облачное хранилище для образов Docker, где пользователи могут как загружать свои собственные образы, так и использовать готовые решения из общедоступного репозитория. Это значительно упрощает процессы обмена и развертывания контейнерных решений.

При работе с системой общение между ее частями выглядит так: человек вводит запрос через панель управления Docker, после чего он поступает к основному серверу Docker посредством специального программного соединения. Сервер принимает запрос, выполняет нужные манипуляции и отправляет результат выполнения обратно через терминал управления.

Применение технологий, связанных с контейнеризацией, при создании и обслуживании программных решений дает множество возможностей. Особенно важно отметить способность к переносу: такие технологии позволяют объединить само приложение вместе со всеми необходимыми элементами в единый образ. Это делает возможным запуск на любом компьютере с поддержкой контейнеризации. Такой подход обеспечивает неизменность условий выполнения программы на каждом этапе - от создания до практического использования, что значительно упрощает перемещение программного обеспечения между разными платформами и системами.

Разделение приложений на уровне процессов и файловой структуры дает возможность одновременно запускать несколько программных решений на одном сервере, исключая вероятность их взаимодействия или конфликтов. Это свойство особенно актуально при работе с архитектурой, основанной на микросервисах. Возможность масштабирования таких упакованных приложений делает их незаменимыми для облачных и распределенных вычислительных сетей. Благодаря этому можно оперативно адаптировать количество работающих экземпляров под конкретные потребности и текущую нагрузку. Кроме того, экономичное использование системных ресурсов считается еще одним важным преимуществом: такие обособленные модули требуют меньших затрат на работу по сравнению с виртуальными машинами, поскольку функционируют на базе единого ядра основной операционной системы хоста.

Однако контейнерные технологии имеют и свои недостатки. Одним из них является безопасность: хотя контейнеры обеспечивают изоляцию, она не так полна, как у виртуальных машин. Уязвимости в ядре хостовой операционной системы могут позволить злоумышленникам выйти за пределы контейнера. Сложность управления в больших масштабах также может стать проблемой, требуя использования сложных оркестрационных инструментов и значительных усилий по мониторингу и поддержке.

Проблемы совместимости могут возникать при запуске контейнеров на разных платформах из-за различий в версиях ядра и библиотек хостовой операционной системы, что требует тщательного тестирования и настройки окружения. В заключение,

контейнерные технологии предоставляют множество преимуществ, таких как портативность, изоляция, масштабируемость и эффективность ресурсов, но также имеют свои недостатки, такие как вопросы безопасности, сложность управления и проблемы совместимости, которые необходимо учитывать.

Технология контейнеризации стала настоящим прорывом в сфере создания и проверки программных продуктов. Возможность объединить само приложение с необходимыми ему компонентами в единый модуль дает разработчикам уверенность в том, что их код будет работать одинаково независимо от среды исполнения. Это эффективно решает проблему совместимости ("у меня работает") и существенно ускоряет процесс создания программ, делая командную работу более удобной. В тестировании контейнеры позволяют мгновенно создавать и удалять изолированные пространства для проверки, что повышает качество и точность тестирования.

В системах непрерывного интегрирования и доставки (CI/CD) роль контейнеров особенно велика. Они обеспечивают автоматизацию всех этапов: от сборки до тестирования и внедрения приложений, гарантируя быстрое и надежное обновление программных решений. Благодаря контейнерам процессы CI/CD становятся более предсказуемыми и контролируемыми, поскольку каждый этап проходит в абсолютно одинаковых условиях. Это значительно снижает вероятность ошибок при переходе от стадии разработки к эксплуатации, ускоряя выпуск новых версий программного обеспечения.

Архитектуры, основанные на микросервисах, получили широкое распространение благодаря своей адаптивности и способности к масштабированию, а контейнеризация только усиливает эти преимущества. Каждый микросервис может быть помещен в отдельный контейнер, что обеспечивает его автономность и защищает от взаимного влияния с другими элементами системы. Такой подход существенно упрощает процессы развертывания, обновления и увеличения мощностей для каждого микросервиса по отдельности, позволяя оперативно адаптироваться к изменяющимся требованиям и нагрузкам. Кроме того, контейнеры делают управление связями между микросервисами более эффективным, что повышает общую стабильность и надежность всей системы.

В сфере облачных вычислений и многокластерных инфраструктур контейнеризация предоставляет продвинутые возможности для управления ресурсами. Благодаря минимальным затратам на запуск по сравнению с виртуальными машинами, контейнеры помогают оптимизировать использование вычислительных мощностей — это особенно важно в условиях облачных решений, где стоимость ресурсов имеет большое значение. Платформы оркестрации, например Kubernetes, предлагают автоматическое масштабирование, балансировку нагрузки и функции самовосстановления контейнеров, что значительно облегчает управление сложными распределенными системами и делает их работу более производительной.

Современная среда контейнерных технологий предлагает разнообразие инструментов, которые делают управление и координацию контейнеров более удобными. Примечательными решениями в этой области являются Docker Compose, Docker Swarm и Kubernetes. Docker Compose позволяет создавать многоконтейнерные приложения с использованием простого YAML-файла, что значительно облегчает их настройку и запуск. Docker Swarm предоставляет встроенные механизмы для управления контейнерами, включая автоматическое масштабирование и администрирование кластеров. Однако лидером среди оркестрационных систем стал Kubernetes, предлагающий широкий спектр возможностей: от автоматического масштабирования до балансировки нагрузки и функций самовосстановления. Сегодня Kubernetes считается фактическим стандартом для управления контейнеризованными приложениями в масштабных и сложных ИТ-средах.

Объединение контейнерных технологий с облачными сервисами, такими как AWS, Google Cloud и Azure, расширяет возможности создания и внедрения программного

обеспечения. Облачные провайдеры предлагают специализированные услуги, оптимизированные для работы с контейнерами, что помогает максимально использовать преимущества облачных вычислений. Например, такие сервисы, как Amazon ECS и EKS, Google Kubernetes Engine и Azure Kubernetes Service, предоставляют готовые платформы для управления контейнерами в облаке, существенно упрощая их развёртывание и адаптацию под изменяющиеся потребности. Это позволяет компаниям оперативно реагировать на рыночные тренды и эффективно распределять ресурсы [5].

Безопасность контейнеров является одной из ключевых задач при их использовании. Основные угрозы и уязвимости включают в себя возможность несанкционированного доступа к контейнерам, эксплуатацию уязвимостей в образах контейнеров и неправильную конфигурацию сетевых политик. Для минимизации рисков рекомендуется следовать лучшим практикам по обеспечению безопасности контейнеров. Среди них можно выделить использование минимальных базовых образов, регулярное обновление и сканирование образов на наличие уязвимостей, ограничение прав доступа к контейнерам и использование изолированных сетевых пространств. Также важно внедрять мониторинг и логирование для своевременного обнаружения и реагирования на инциденты безопасности.

Перспективы развития контейнерных технологий сулят множество инноваций и тенденций, которые могут кардинально изменить подход к созданию и управлению программными решениями. Одной из основных направлений является активное внедрение серверлесс-архитектур, где контейнеры становятся ключевым элементом для реализации функционала "по запросу". Параллельно наблюдается рост интереса к мультикластерным системам, позволяющим координировать работу контейнеров в гибридных средах — как облачных, так и локальных. Это обеспечивает высокую отказоустойчивость и адаптивность. Кроме того, использование технологий машинного обучения и искусственного интеллекта для автоматизации процессов оркестрации открывает новые горизонты оптимизации и управления ИТ-инфраструктурами.

Среди возможных направлений эволюции стоит отметить совершенствование инструментов безопасности, разработку новых стандартов взаимодействия между контейнерами и расширение поддержки различных типов аппаратных архитектур. Эти изменения способны существенно улучшить характеристики современных ИТ-систем, сделав их более надежными, защищенными и производительными. Контейнерные технологии продолжают развиваться, предлагая все более совершенные решения для создания, внедрения и администрирования приложений в условиях постоянно меняющихся требований современного мира.

Список литературы:

1. Бикбулатов, Р. И. Оценка актуальности и эффективности использования интеллектуальных систем в логистической сфере / Р. И. Бикбулатов, О. В. Борисова // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2024. – № 1. – С. 24-27. – EDN GWDJER.
2. Трофимов, В. В. Использование виртуализации и контейнерной технологии для формирования информационной инфраструктуры / В. В. Трофимов, В. И. Кияев, С. М. Газуль // Международная конференция по мягким вычислениям и измерениям. – 2017. – Т. 2. – С. 348-351.
3. Абрамов, И. В. Использование Docker для виртуализации инфраструктуры распределенных вычислений / И. В. Абрамов // Радиоэлектроника, электротехника и энергетика : тезисы докладов Двадцать второй Международной научно-

технической конференции студентов и аспирантов: в 3-х томах, Москва, 25–26 февраля 2016 года. Том 1. – Москва: Издательский дом МЭИ, 2016. – С. 240.
4. Емельянова, С. С. Исследование Docker в части контейнеризации приложений на уровне ос / С. С. Емельянова, Н. Н. Иващенко // Научно-технический вестник Поволжья. – 2021. – № 12. – С. 149-151.
5. Виртуализация сетевых функций в программно-конфигурируемых сетях / А. Н. Ганиев, А. Н. Григорчук, Б. П. Репин, Б. Г. Репин // Научная мысль. – 2021. – Т. 15, № 1-1(39). – С. 46-51.

References:

1. Bikbulatov, R. I. Assessment of the relevance and effectiveness of the use of intelligent systems in the logistics sector / R. I. Bikbulatov, O. V. Borisova // Modern science: actual problems of theory and practice. Series: Natural and Technical Sciences. – 2024. – No. 1. – PP. 24-27. – EDN GWDJER.
2. Trofimov, V. V. For the use of the virtual infrastructure of container information technology for the formation of a year / V. V. Trofimov, V. Year. Kiyaeva, S. M. Gazelle // International conference boiler of non–measurement of soft computing. – 2017. – Vol. 2. - pp. 348-351.
3. Abramov, Year. V. Distributed computing of virtual infrastructure for using Docker / year. V. Abramov // Radio electronics, electrical engineering, power engineering of the year: abstracts of the international scientific School of Engineering and Postgraduate student conferences of the year: in 3 Volumes, Moscow, February 25-26, 2016 rustle. Volume 1. Moscow: Publishing house dota MEI, 2016. p. 240.
4. Yemelyanov, S. S. The Docker container is often trying to level research in the application of V. / S. S. Yemelyanova, N. N. Ivashchenko // Scientific and Technical Bulletin of the Volga region. - 2021. – No. 12. – pp. 149-151.
5. Network virtualization functions, V. software and configurable networks / A. N. Ganiev, A. N. Grigorchuk, B. Very. Repina, B. G. Repina // Scientific thought. - 2021. – Vol. 15, No. 1-1(39). – pp. 46-51.