Приложения — в контейнеры. Угрозы — за борт

Как сделать разработку с использованием контейнеров безопасной. Краткое руководство

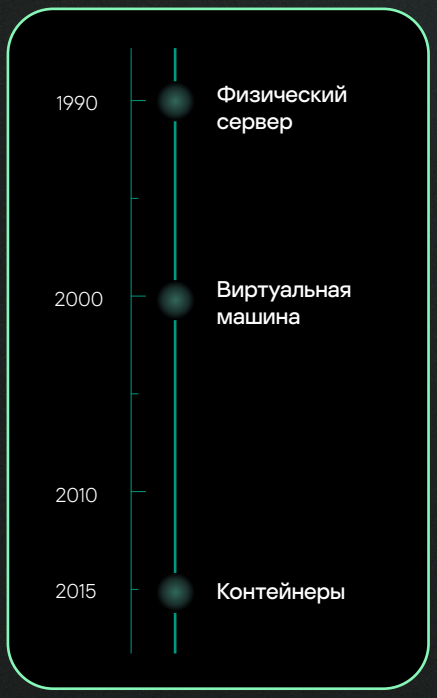
kaspersky

Содержание

1. Что такое контейнер?

Контейнер — это легкий программный компонент, внутри которого расположена среда, необходимая для работы приложения (микросервиса): файлы, библиотеки и метаданные. В отличие от виртуальной машины, не имеет собственной ОС, а работает на операционной системе, установленной на узле. На текущий момент контейнеризация приложений — технология, обеспечивающая наиболее удобное развертывание ПО и эффективное использование вычислительных ресурсов.

Этапы развития технологий виртуализации



Физические серверы

Архитектура «клиент-сервер» реализуется на базе физических серверов, на каждом из которых используется одна операционная система, зачастую обслуживающая только одно приложение. Экспоненциальный рост числа приложений привел к постоянным проблемам с распределением ресурсов физических серверов.

Виртуальные машины

Возможность запуска в программном обеспечении для виртуализации нескольких экземпляров операционной системы (ОС) на одном сервере устранила зависимость вычислительной среды от физической инфраструктуры. Благодаря этому аппаратные ресурсы одного компьютера могут использоваться совместно несколькими изолированными средами.

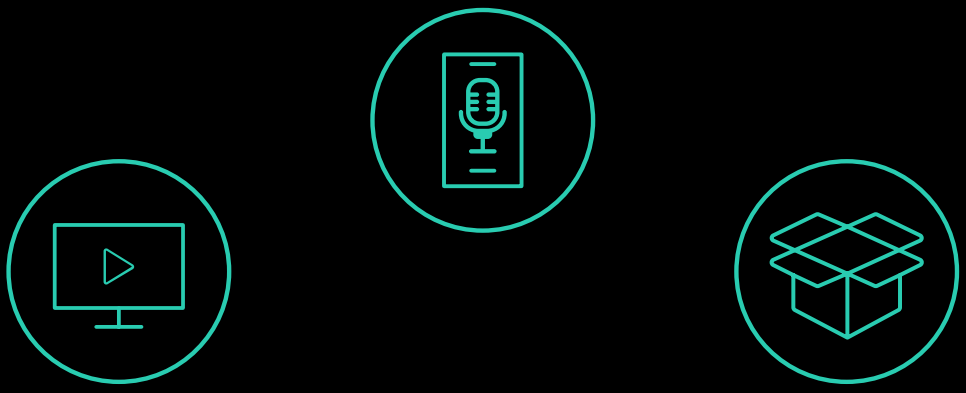
Контейнеры

С появлением контейнеров виртуализация перешла на новый уровень: стало возможным запускать приложения в выделенных средах в одной и той же операционной системе, на одной виртуальной машине или на одном сервере. По прогнозам Gartner, уровень внедрения Kubernetes (приложение для оркестрации контейнеров) вскоре превысит 75%[1].

2. Повседневное использование контейнеров

Контейнеры нашли широкое применение в приложениях, построенных с использованием микросервисной архитектуры. Каждый микросервис в подобном приложении — отдельный и независимый от других частей кода бизнес-процесс. За счет этого его легко поместить в контейнер, чтобы облегчить развертывание всего приложения (продукта) и повысить его надежность.

Будь то приложение вашего любимого музыкального сервиса, службы доставки или же сложные промышленные решения — все они, скорее всего, построены с использованием контейнеров.

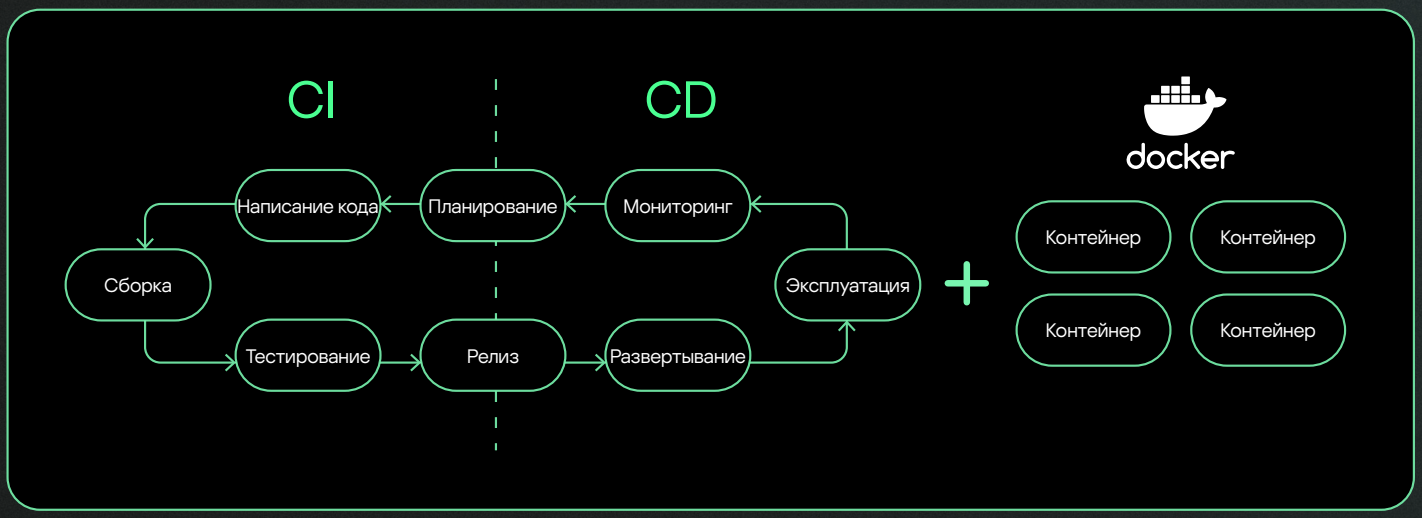


3. Жизненный цикл разработки программного обеспечения (CI/CD) и контейнеры

Принцип непрерывной интеграции и доставки (CI/CD) стал новым стандартом в разработке ПО. По данным исследования, 83% российских IT-компаний уже перешли на CI/CD[2].

Непрерывная интеграция (Continuous integration, CI) — это методология написания кода, позволяющая автоматизировать сборку и тестирование приложений.

Принцип непрерывной доставки (Continuous delivery, CD) позволяет автоматизировать внесение изменений в код при разработке, тестировании и развертывании приложений. При таком подходе команды разработчиков могут непрерывно и параллельно работать над приложением и постоянно модифицировать его код. Это повышает как эффективность совместной работы, так и качество кода.



Как использование контейнеров дополняет преимущества CI/CD

Стандартизация — возможность запуска в любой среде.

Низкое потребление ресурсов — использование одного и того же ядра системы ОС .

Безопасность — код микросервиса изолирован от среды.

Контейнеризация (помещение программного компонента и его окружения, зависимостей и конфигурации в контейнер) позволяет ускорить разработку и доставку приложений, многократно увеличивая преимущества цикла CI/CD:

- ускоряет написание кода, отладку и релиз;
- повышает стабильность работы приложения;
- снижает требования к инфраструктуре для разработчика и заказчика;
- легко масштабируется для реализации любых проектов;
- повышает надежность сборки за счет гибкой оркестрации контейнерных кластеров.

4. Компоненты контейнерной инфраструктуры

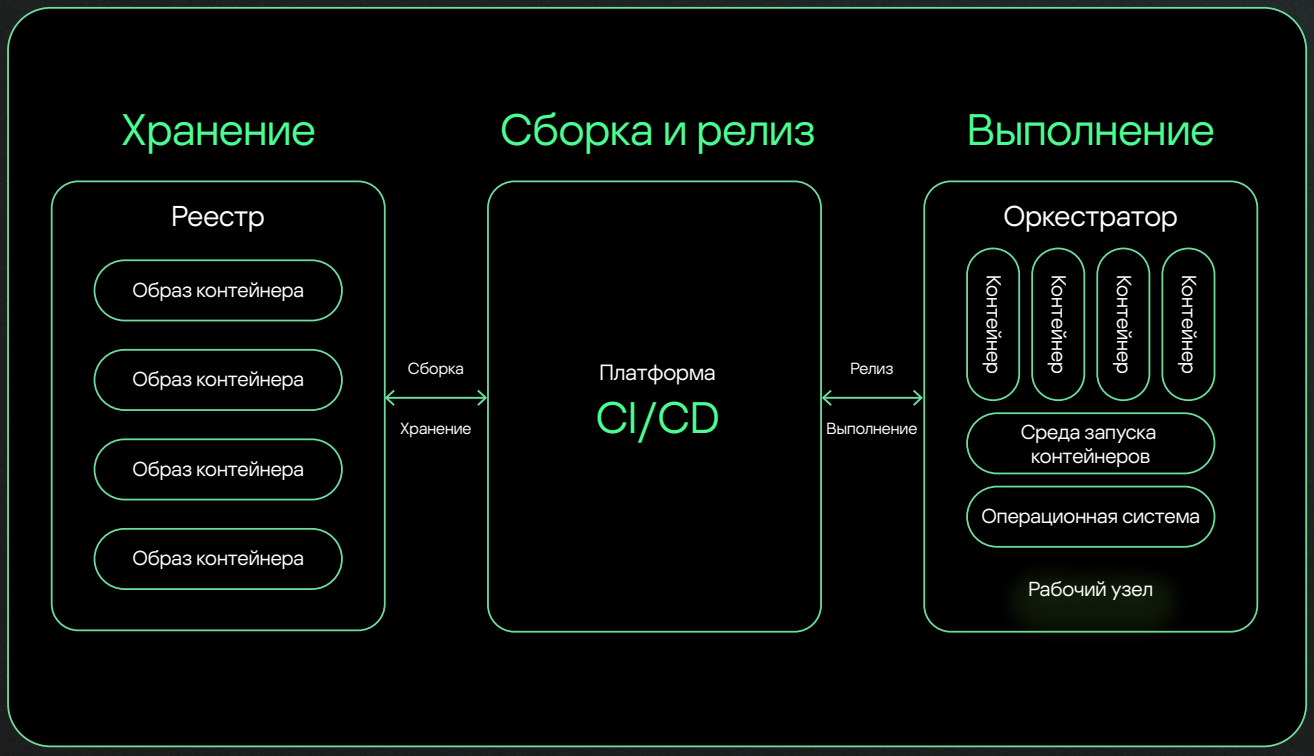
Образ. Основной компонент контейнерной архитектуры — статический файл с кодом, базовой конфигурацией и компонентами. Шаблон, из которого создается контейнер. Образ существует отдельно и не изменяется. Создается клиентами (разработчиками) самостоятельно или располагается в сторонних реестрах образов, например в Docker Hub.

Реестр образов. Хранилище образов контейнеров, которое используется в рамках Ci/CD-процессов при разработке контейнерных приложений. Наиболее популярными реестрами образов являются Docker Hub, JFrog, Sonatype Nexus OSS, GitLab Registry, Harbor. Существуют реестры с открытым и закрытым доступом.

Оркестратор. Инструмент для развертывания и управления большим количеством контейнеров, позволяющий разработчикам использовать кластеры контейнеров для масштабирования и автоматизировать процесс объединения контейнеров в группы. Наиболее популярными являются Kubernetes, OpenShift и Docker Swarm.

Контейнер. Легкий программный компонент, внутри которого расположена среда, необходимая для работы приложения (микросервиса): файлы, библиотеки и метаданные. В отличие от виртуальной машины, не имеет собственной ОС, а работает на операционной системе, установленной на узле.

Операционная система (ОС). Основная программа, управляющая всеми остальными прикладными программами на компьютере.



5. Ключевые риски безопасности контейнеров

респондентов отмечают снижение доходов и (или) сокращение числа клиентов в результате инцидентов, связанных с безопасностью контейнеров.[3]

компаний столкнулись как минимум с одним инцидентом при использовании Kubernetes за последние 12 месяцев.[3]

Образы

- Открытые внешние источники
- Уязвимости программного обеспечения
- Ошибки в конфигурации
- Вредоносное ПО
- Конфиденциальная информация в открытом доступе
- Использование ненадежных образов

Реестр образов

- Незащищенные соединения
- Наличие устаревших образов с уязвимостями и вредоносным ПО
- Неэффективные/ непродуманные политики авторизации и аутентификации

Оркестратор

- Неограниченный административный доступ
- Доступ без авторизации
- Слабое разделение трафика между контейнерами или его отсутствие
- Контейнеры с различными уровнями защиты данных не разнесены по разным хостам
- Ошибка в конфигурации оркестратора

Контейнеры

- Уязвимости среды выполнения
- Неограниченный доступ контейнеров к сети
- Небезопасные конфигурации
- Уязвимости приложений в контейнерах
- Незапланированные контейнеры в среде выполнения

ОС хоста

- Большая поверхность атаки
- Общее ядро ОС для всех контейнеров
- Уязвимости в компонентах ОС
- Неправильная настройка прав доступа пользователей
- Возможность доступа контейнеров к файловой системе

Примеры инцидентов, связанных с безопасностью контейнеров*

	Инцидент	Результат	Компания
Образы	Размещение зараженных образов злоумышленниками в свободном доступе	До момента устранения уязвимости (через 30 дней) с помощью образов было похищено порядка 90 тыс. долл. США	Популярный реестр
Реестр образов	Размещение реестра образов в открытом доступе	Риск использования уязвимости для получения доступа к персональным данным	Авиалинии
Оркестратор	Проникновение в систему через незащищенную консоль управления K8s	• Майнинг криптовалюты на серверах • Утечка конфиденциальной информации	Производитель автомобилей
Контейнеры	Проникновение с использованием неприватного API и запуск контейнера с привилегированными правами	Получение доступа к любому устройству в сети	Исследовательская компания
ОС хоста	Проникновение в систему путем назначения административных прав хосту Docker	Получение доступа к бессерверным системам	ИБ-компания по обеспечению кибербезопасности

* На основе общедоступных данных

6. Безопасность: контейнеры и виртуальные машины

Традиционные решения по обеспечению безопасности защищают виртуальные машины, но не контейнерные платформы.

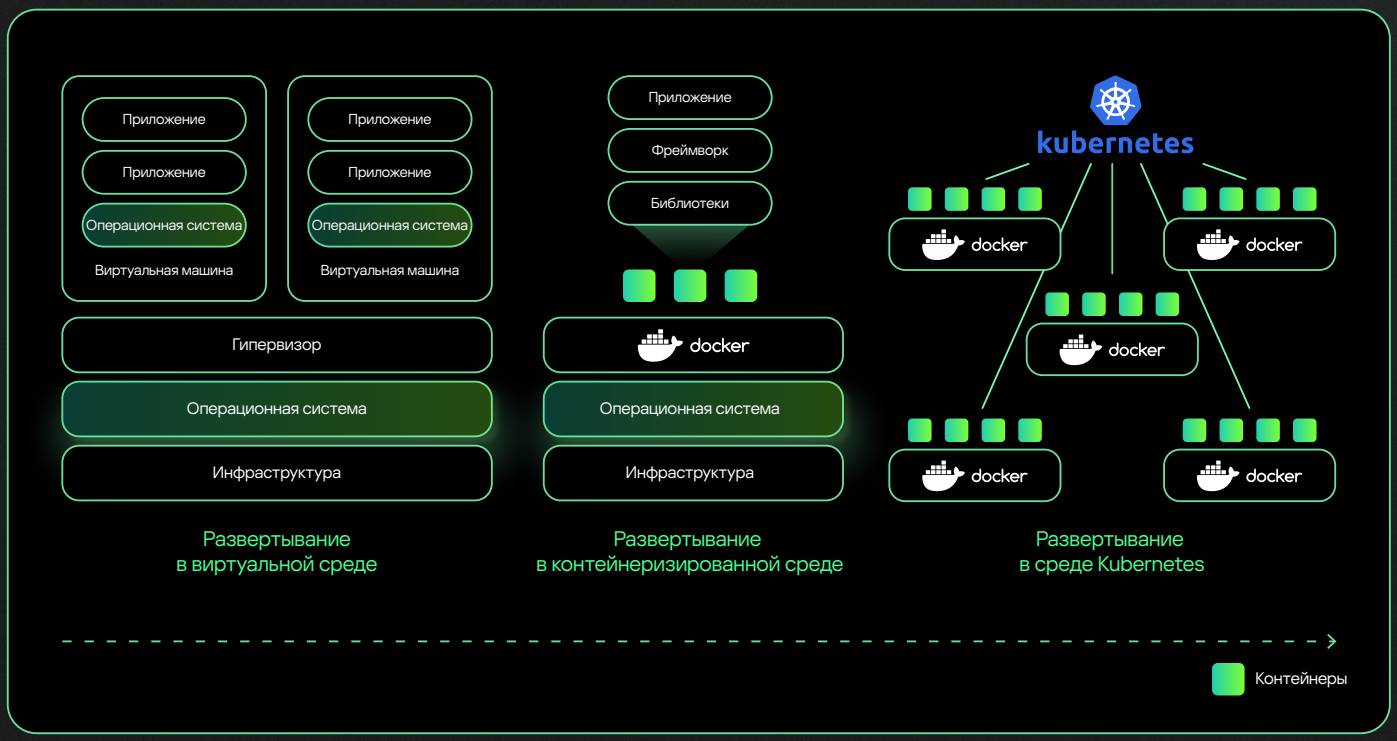
Защита виртуальной машины принципиально не отличается от защиты физического компьютера. Агент безопасности легко установить в ОС виртуальной машины.

В отличие от виртуальных машин, контейнеры не имеют собственной ОС и используют ОС хоста, поэтому установить агент безопасности виртуальной машины в контейнер невозможно. Как следствие, традиционные решения имеют серьезные ограничения в части обеспечения безопасности контейнеров, в том числе:

- отсутствие анализа и контроля ошибок в конфигурации оркестратора;
- отсутствие интеграции с процессами разработки, реестрами образов, CI/CD и оркестраторами;
- отсутствие визуализации компонентов контейнерных сред.

Почему для контейнерной архитектуры нужно специализированное решение:

- Контейнеры используют ОС хоста. Уязвимости безопасности могут присутствовать в каждом из компонентов контейнерной инфраструктуры: образах, реестрах образов, оркестраторах, контейнерах и ОС хоста.
- Скомпрометированный контейнер открывает путь для атаки на все контейнеры, использующие общую ОС хоста. Скомпрометированная ОС хоста открывает путь для атаки на все использующие ее контейнеры.
- Решения для обеспечения безопасности виртуальных машин не обеспечивают защиту и не выявляют проблемы на ранних этапах работы с контейнерами — во время хранения и разработки.
- Решения для обеспечения безопасности виртуальных машин не обеспечивают анализ и контроль ошибок в конфигурации оркестратора.
- Внедрение защитных решений значительно отстает от бурного роста применения контейнерных технологий.



7. Kaspersky Container Security

Kaspersky Container Security — это специализированное решение для защиты контейнерных сред, которое поможет снизить распространенные риски информационной безопасности. Интуитивно понятный интерфейс позволит быстро освоить решение даже неподготовленному пользователю, а низкое потребление вычислительных ресурсов делает его доступным для использования практически в любой инфраструктуре.

- Защита всех компонентов контейнерной инфраструктуры: образов, реестров образов, оркестраторов, контейнеров, ОС хоста.
- Легкая интеграция в процессы безопасной разработки ПО.
- Защита на этапах хранения и сборки, а также во время выполнения.
- Автоматизация анализа конфигурации и проверок на соответствие требованиям.

Решение Kaspersky Container Security состоит из трех компонентов, которые комплексно обеспечивают безопасность контейнерных сред.

Агент Kaspersky Container Security

- Обнаруживает уязвимости на уровне контейнеров, кластеров и оркестратора, обеспечивая безопасность среды выполнения.
- Устанавливается в кластер в виде отдельного контейнера.

Сканер образов и инфраструктуры Kaspersky Container Security

- Проверяет реестр образов на актуальность и безопасность.
- Проверяет образы в рамках CI-процесса и снижает риски на этапе сборки.
- Устанавливается в кластер с серверными компонентами оркестратора.

KCS Control Server

- Мониторит состояния компонентов решения и взаимодействия между ними, а также собирает информацию о выявленных событиях.
- Устанавливается в кластер с серверными компонентами оркестратора.
- Контролирует доставку образов и аудит соблюдения политики безопасности.
- Интегрируется с SIEM и другими решениями сторонних производителей.

Интеграция в процесс разработки

Система контроля версий и реестр

Исследование

Проверка образов из реестра

Сканирование конфигурационных файлов (IaC, Dockerfile) на наличие ошибок и секретов

Инструменты CI

Написание кода и тестирование

Сканирование образов на уязвимости, вредоносное ПО и наличие секретов

Инструменты CD

Доставка и развертывание

Доставка образов, соответствующих политикам безопасности

Оркестратор

Выполнение

Проверка сетевого взаимодействия на соответствие политикам безопасности

Поведенческий анализ контейнеров

Преимущества Kaspersky Container Security

- Оптимизация затрат на надежную защиту контейнерной среды.
- Простая эксплуатация, снижающая нагрузку на службу информационной безопасности.
- Полнофункциональный продукт, соответствующий лучшим мировым практикам в области контейнерной безопасности.
- Соответствие нормативным требованиям и отраслевым стандартам безопасности; интеграция с SIEM и другими решениями Лаборатории Касперского и сторонних производителей.
- Создан для российского рынка: анализ по БДУ (ФСТЭК) и поддержка отечественных ОС Astra Linux и РЕД ОС.

Снижение рисков, связанных с использованием ключевых компонентов в контейнеризированных средах

Образы

Проверка на наличие уязвимостей

Проверка ошибок в конфигурациях образов

Проверка на наличие вредоносного ПО

Проверка на наличие конфиденциальной информации

Оценка рисков и выявление потенциально опасных образов

Реестр образов

Интеграция с реестрами и валидация образов согласно требованиям политик проверки

Использование актуальных безопасных образов

Оркестратор

Обнаружение ошибок конфигурации и предоставление рекомендаций по их исправлению

Визуализация ресурсов в кластере

Мониторинг трафика

Обнаружение и проверка образов в кластере

Контейнеры

Контроль запуска и работы только доверенных контейнеров

Мониторинг трафика

Контроль целостности контейнеров

Защита от файловых угроз

Поведенческий анализ

Контроль работы приложений и сервисов внутри контейнеров

ОС хоста

Проверка версий компонентов базовой ОС

Обнаружение ошибок конфигурации и предоставление рекомендаций по исправлению

Снижение рисков за счет мониторинга контейнеров

1 What Is Kubernetes And When to Use It: Key Trends in 2023 // Medium.com, 13 декабря 2022 г.

2 «Исследование состояния DevOps в России 2023» // «Экспресс 42», июль 2023 г.

3 Kubernetes adoption, security, and market trends report 2023 // Red Hat. 17 апреля 2023.

Kaspersky Container Security

Узнать больше

www.kaspersky.ru