Министерство образования Республики Молдова

Технический университет Молдовы

**РЕФЕРАТ**

**‹‹ Веб - технологии ››**

**Тема: «Docker»**

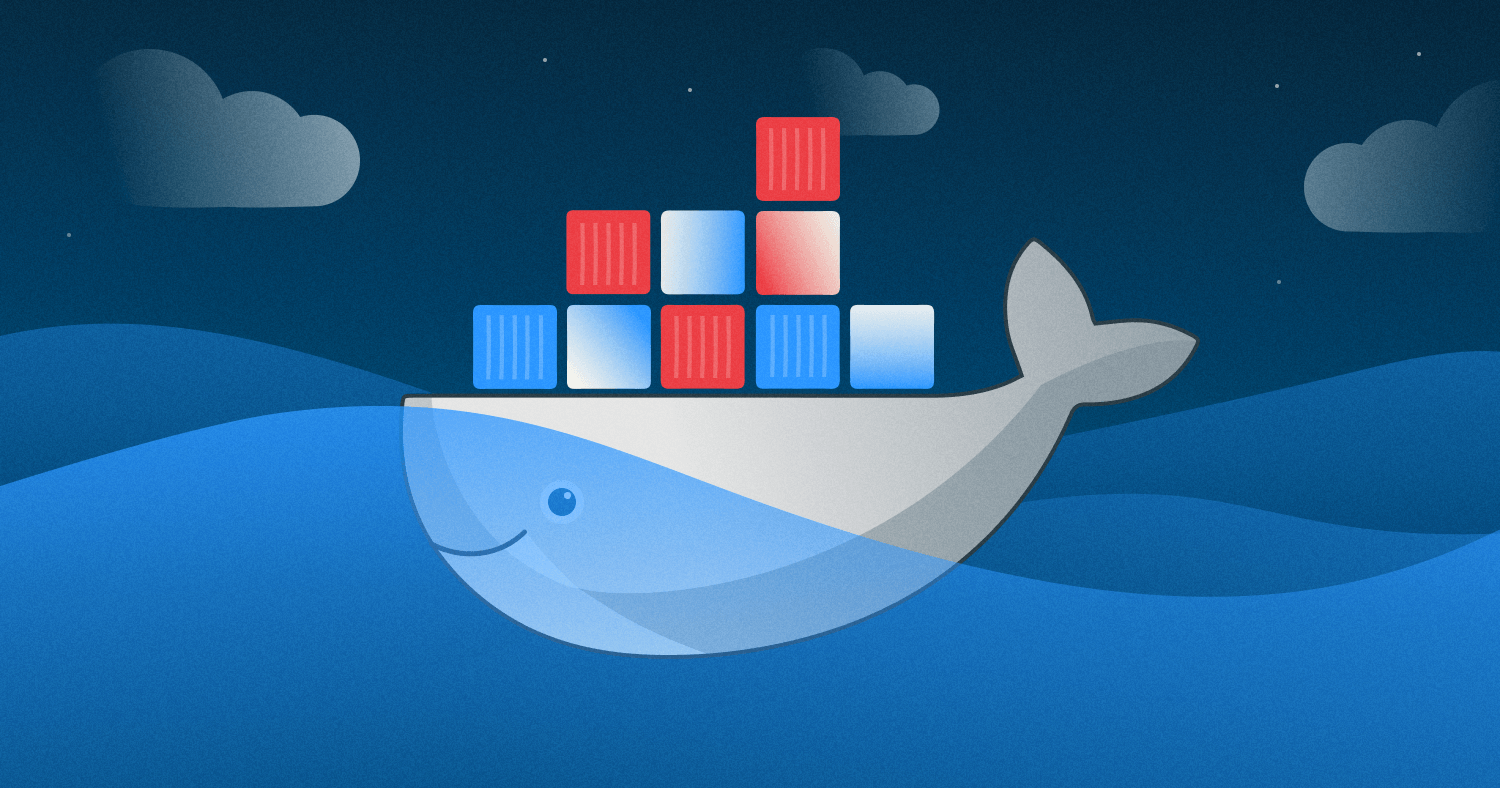
Выполнил: ст. гр. CR-223

Пистрюга Александр

Проверил: Bargan Constantin

Кишинев – 2024

Что такое контейнеры



Прежде чем рассказывать про Docker, нужно сказать несколько слов о технологии контейнеризации.

**Контейнеры** — это способ стандартизации развертки приложения и отделения его от общей инфраструктуры. Экземпляр приложения запускается в изолированной среде, не влияющей на основную операционную систему.

Разработчикам не нужно задумываться, в каком окружении будет работать их приложение, будут ли там нужные настройки и зависимости. Они просто создают приложение, упаковывают все зависимости и настройки в некоторый единый образ. Затем этот образ можно запускать на других системах, не беспокоясь, что приложение не запустится.

**Docker** — это платформа для разработки, доставки и запуска контейнерных приложений. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

Контейнеризация похожа на виртуализацию, но это не одно и то же. Виртуализация запускает полноценный хост на гипервизоре со своим виртуальным оборудованием и операционной системой. При этом внутри одной ОС можно запустить другую ОС. В случае контейнеризации процесс запускается прямо из ядра основной операционной системы и не виртуализирует оборудование. Это означает, что контейнеризованное приложение может работать только в той же ОС, что и основная. Контейнеры не виртуализируют оборудование, поэтому потребляют меньше ресурсов.

Преимущества использования контейнеров Docker

Контейнеры упрощают работу как программистам, так и администраторам, которые развертывают эти приложения.

Docker решает проблемы зависимостей и рабочего окружения

Контейнеры позволяют упаковать в единый образ приложение и все его зависимости: библиотеки, системные утилиты и файлы настройки. Это упрощает перенос приложения на другую инфраструктуру.

Например, разработчики создают приложение в системе разработки — там все настроено, приложение работает. Когда оно готово, его нужно перенести в систему тестирования, а затем в продуктивную среду. Если в одной из них нет нужной зависимости, приложение не будет работать. Программистам придется отвлечься от разработки и совместно с командой поддержки разобраться в ситуации.

В контейнерах такой проблемы нет, так как они содержат в себе все необходимое для запуска приложения. Специалисты занимаются разработкой, а не решением инфраструктурных проблем.

Изоляция и безопасность

Контейнер — это набор процессов, изолированных от основной операционной системы. Приложения работают только внутри контейнеров и не имеют доступа к основной операционной системе. Это повышает безопасность приложений:они не смогут случайно или умышленно навредить основной системе. Если приложение в контейнере завершится с ошибкой или зависнет, это никак не затронет основную ОС.

Ускорение и автоматизация развертывания приложений и масштабируемость

Контейнеры упрощают развертывание приложений. В классическом подходе для установки программы нужно совершить несколько действий: выполнить скрипт, изменить файлы настроек и так далее. В этом процессе не исключена вероятность человеческой ошибки: пользователь запустит скрипт два раза, перепутает последовательность или что-то не поймет. Контейнеры позволяют полностью автоматизировать этот процесс, так как включают в себя все нужные зависимости и порядок выполнения действий.

Также контейнеры упрощают развертывание на нескольких серверах. В классическом подходе для того, чтобы развернуть одно и то же приложение на нескольких машинах, нужно будет повторять одни и те же действия. Контейнеры избавляют от этой рутинной работы и позволяют автоматизировать развертывание.

Контейнеры приближают к микросервисной архитектуре

Контейнеры хорошо вписываются в микросервисную архитектуру. Это подход к разработке, при котором приложение разбивается на небольшие компоненты, по возможности независимые. Обычно противопоставляется монолитной архитектуре, где все части системы сильно связаны друг с другом.

Это позволяет разрабатывать новую функциональность быстрее, ведь в случае с монолитной архитектурой изменение какой-то части может затронуть всю остальную систему.

Docker compose — одновременно развернуть несколько контейнеров

Docker-compose позволяет разворачивать и настраивать несколько контейнеров одновременно. Например, для веб-приложения нужно развернуть стек LAMP: Linux + Apache, MySQL, PHP. Каждое из приложений — это отдельный контейнер для ОС Linux. Но в этой ситуации нам нужны именно все контейнеры вместе, а не отдельно взятое приложение. Docker-compose позволяет развернуть и настроить все приложения одной командой, а без него пришлось бы разворачивать и настраивать каждый контейнер отдельно.

Хранение данных в Docker

Одна из главных особенностей контейнеров — эфемерность. Это означает, что контейнеры могут быть в любой момент остановлены, перезапущены или уничтожены. При этом все накопленные данные в контейнере будут потеряны. Поэтому приложения нужно разрабатывать так, чтобы они не полагались на хранилище данных в контейнере, это называется принципом Stateless.

Это хорошо подходит для приложений или сервисов, которые не сохраняют результаты своей работы. Например, функции расчета или преобразования данных: им на вход поступил один набор данных, они его преобразовали или рассчитали и вернули результат. Все, ничего никуда сохранять не нужно.

Но далеко не все приложения такие, и есть много данных, которые нужно сохранить. В контейнерах для этого предусмотрены несколько способов.

Тома (Docker volumes)

Это способ, при котором Docker сам создает директории для хранения данных. Их можно сделать доступными для разных контейнеров, чтобы они могли обмениваться данными. По умолчанию эти директории создаются на хост-машине, но можно использовать и удаленные хранилища: файловый сервер или [объектное хранилище](https://slc.tl/I1MVZ).

Монтирование каталога (bind mount)

В этом случае директория сначала создается на хост-машине а уже потом монтируется в контейнеры.

Но этот способ не рекомендуется, потому что он усложняет резервное копирование, миграцию и совместное использование данных несколькими контейнерами.

Архитектура (компоненты) Docker

Теперь расскажем подробнее про компоненты, из которых состоит Docker.

Docker daemon

Это некоторый резидентный процесс, который запущен на хост-машине постоянно. Он владеет всей инфраструктурой, а также предоставляет интерфейс взаимодействия с контейнерами, включающего создание и удаление, запуск и остановку.

В ранних версиях платформы Docker можно встретить упоминание о dockerd, но на текущий момент демоны уже успели разбиться на отдельные проекты. Все чаще можно встретить его современника — containerd.

Docker client (клиент)

Это интерфейс командной строки для управления Docker daemon. Мы пользуемся этим клиентом, когда создаем и разворачиваем контейнеры, а клиент отправляет эти запросы в Docker daemon.

Docker image (образ)

Это неизменяемый файл (образ), из которого разворачиваются контейнеры. Приложения упаковываются именно в образы, из которых потом уже создаются контейнеры. В технической литературе можно также встретить описание image как шаблона запуска процесса.

Приведем аналогию на примере установки операционной системы. В дистрибутиве (образе) ОС есть все, что необходимо для ее установки. Но этот образ нельзя запустить, для начала его нужно «развернуть» в готовую ОС. Так вот, дистрибутив для установки ОС — это образ, а установленная и работающая ОС — это контейнер. Но контейнеры обычно разворачиваются одной командой — это намного проще и быстрее, чем установка ОС.

Docker container (контейнер)

Это уже развернутое из образа и работающее приложение.

Docker Registry

Это репозиторий с образами. Разработчики создают образы своих программ и выкладывают их в репозиторий, чтобы их можно было скачать и воспользоваться ими. Распространенный публичный репозиторий — [Docker Hub](https://hub.docker.com/" \t "_blank). В нем собраны образы множества популярных программ или платформ: базы данных, веб-серверы, компиляторы, операционные системы и так далее. Также можно создать свой приватный репозиторий, например внутри компании. Разработчики будут размещать там образы, которые будут использоваться всей компанией.

Docker, по порядку, делает следующее:

* скачивает образ ubuntu: docker проверяет наличие образа ubuntu на локальной машине, и если его нет — то скачивает его с Docker Hub. Если же образ есть, то использует его для создания контейнера;
* создает контейнер: когда образ получен, docker использует его для создания контейнера;
* инициализирует файловую систему и монтирует read-only уровень: контейнер создан в файловой системе и read-only уровень добавлен образ;
* инициализирует сеть/мост: создает сетевой интерфейс, который позволяет docker-у общаться хост машиной;
* Установка IP адреса: находит и задает адрес;
* Запускает указанный процесс: запускает ваше приложение;
* Обрабатывает и выдает вывод вашего приложения: подключается и логирует стандартный вход, вывод и поток ошибок вашего приложения, что бы вы могли отслеживать как работает ваше приложение.

Заключение

Docker - это мощная платформа, которая может революционизировать способ разработки, развертывания и управления приложениями. Благодаря своей простоте использования, портативности и масштабируемости Docker становится все более популярным выбором для разработчиков, системных администраторов и предприятий.