

Использование технологии **Docker** в узлах распределенной системы.

Непрямое взаимодействие в
распределенных системах.

Технология докеризации сервисов распределенной системы

- **Docker** – это открытая платформа, предназначенная для разработки, доставки и организации работы приложений.

Технология докеризации сервисов распределенной системы

- **Docker позволяет** отделить приложения от инфраструктуры так, чтобы доставлять программное обеспечение максимально быстро.
- **При помощи Docker** можно управлять инфраструктурой так же, как и приложениями.
- **Сокращение времени** между написанием кода и его работой.

Платформа Docker

- **Docker** дает возможность упаковать и запустить приложение в слабо изолированном окружении, называемом контейнер.
- **Изоляция и безопасность решения позволяют** запускать множество контейнеров одновременно на нужном хосте.
- Благодаря легковесной природе контейнеров, без дополнительных затрат на гипервизор **можно запускать больше контейнеров** на доступном железе, чем если бы использовались виртуальные машины.

Платформа Docker

➤ **Docker** предоставляет утилиты и платформу для управления жизненным циклом ваших контейнеров:

- **Инкапсулировать** приложения (и поддерживаемые компоненты) в контейнеры **Docker**
- **Распространять и доставлять** эти контейнеры командам для дальнейшей разработки и тестирования
- **Разворачивать** эти приложения в продакт окружении, что бы это ни было, локальный датацентр или облако.

Виртуальные машины vs Docker

- И контейнеры **Docker**, и **виртуальные машины** — это средства для развертывания приложений в окружении, изолированном от железа хоста. Ключевая разница между ними **в уровне изоляции.**

Если речь идет о виртуальной машине (**VM**), то все, что находится внутри нее, не зависит от операционной системы хоста. **VM**-платформа запускает процесс (virtual machine monitor или VMM) для управления процессом виртуализации отдельной виртуальной машины. Система хоста выделяет некоторое количество ресурсов машины для этой **VM**.

VM кардинально отличается тем, что при ее запуске загружается новое выделенное ядро для ее среды и запускается набор процессов операционной системы (часто довольно большой). Это делает размер виртуальной машины намного больше размера типичного контейнера, содержащего только приложение.

Виртуальные машины vs Docker

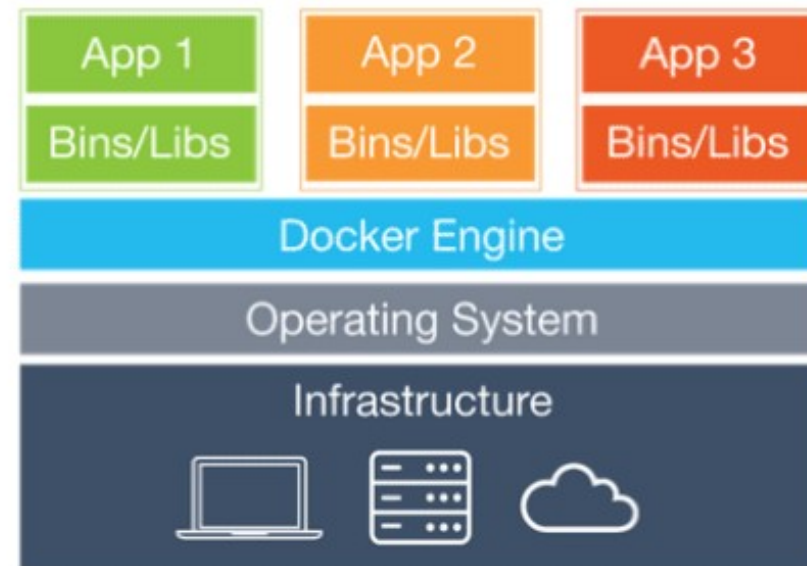
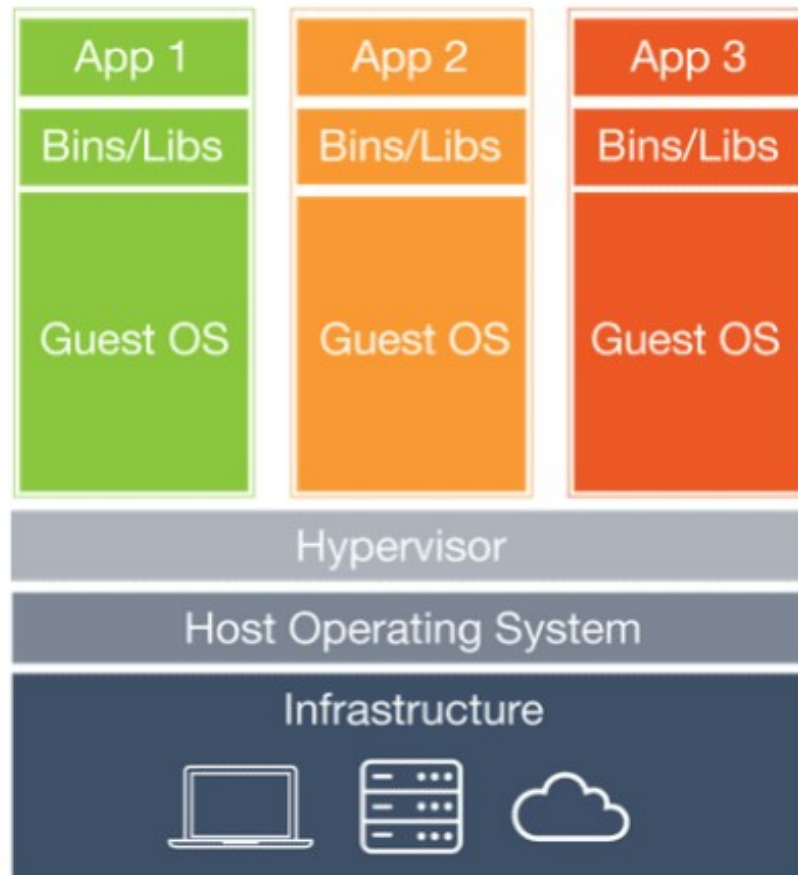
- И контейнеры **Docker**, и **виртуальные машины** — это средства для развертывания приложений в окружении, изолированном от железа хоста. Ключевая разница между ними **в уровне изоляции.**

В контейнерной среде запуска вроде **Docker** ваше приложение помещается в песочницу с изолированными функциями, предоставляемыми контейнером. При этом оно делит ресурсы ядра с другими приложениями в контейнерах на этом же хосте.

В результате процессы, происходящие внутри контейнеров, видимы из системы хоста. Конечно, если у вас достаточно прав, чтобы их видеть.

Контейнеры не нуждаются в отдельной операционной системе, они очень легковесны, обычно это 5-100 MB.

Виртуальные машины vs Docker

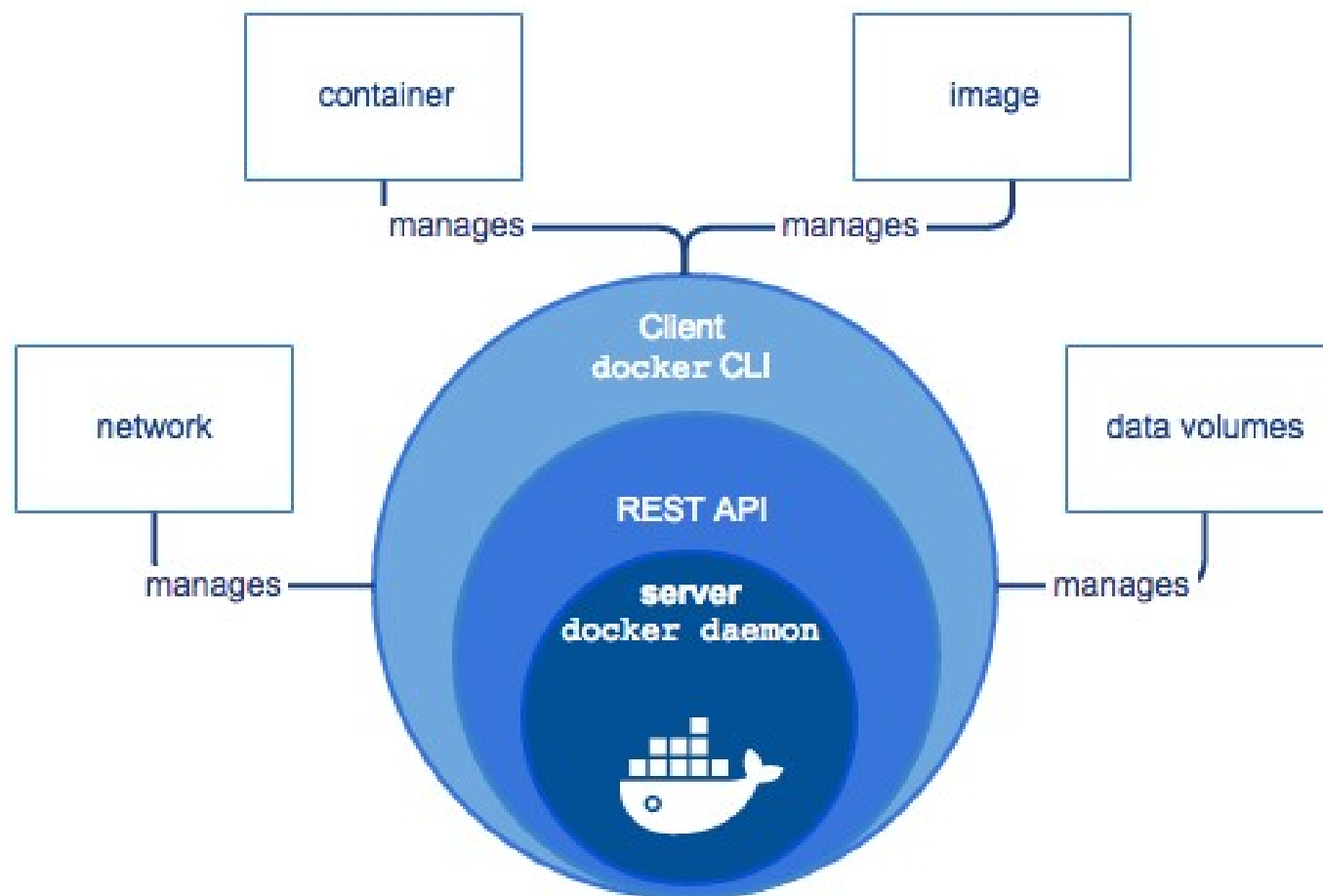


Docker Engine

Docker Engine — это клиент-серверное приложение, содержащее следующие основные компоненты:

- **Сервер**, по сути являющийся долго работающим приложением, называемым демоном.
- **REST API**, определяющее интерфейсы для взаимодействия, которые могут использовать другие программы.
- **Консольный (CLI) клиент.**

Docker Engine



Для чего использовать Docker?

Быстрая, последовательная доставка ваших приложений

Docker может упростить жизненный цикл разработки, позволяя разработчикам работать в стандартизованных окружениях, используя локальные контейнеры, в которых может работать ваше приложение или сервисы. Вы также можете интегрировать **Docker** в ваши процессы непрерывной интеграции (continuous integration) и непрерывной доставки (continuous deployment).

Для чего использовать Docker?

Адаптивная доставка и масштабируемость

Используя контейнеры платформа **Docker** позволяет хорошо переносить нагрузки. Контейнеры **Docker** могут работать на локальной машине разработчика, на физическом или виртуальном сервере в датацентре, в Облаке, или в смешанном окружении.

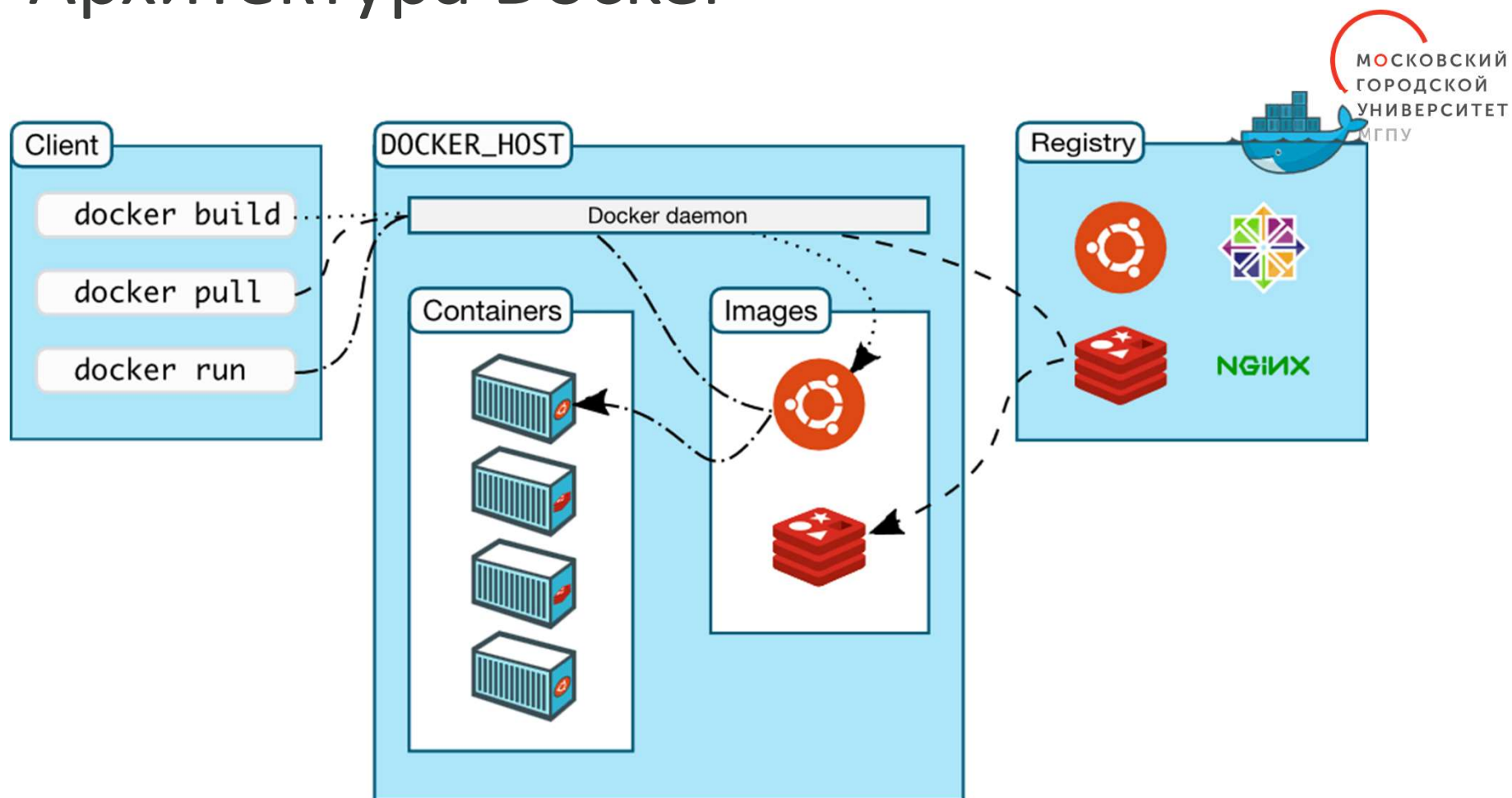
Переносимость и легковесная природа **Docker** также позволяют просто управлять нагрузкой, масштабировать приложения и сервисы вверх или вниз настолько быстро, насколько этого требует бизнес, почти в реальном времени.

Для чего использовать Docker?

Обслуживание больших нагрузок на том же самом железе

Docker легковесен и очень быстр. Он предоставляет жизнеспособную и экономически эффективную альтернативу виртуальным машинам, запускаемым в гипервизорах, позволяя вам использовать больше ваших вычислительных мощностей для достижения ваших бизнес целей. Это особенно важно при использовании его в очень уплотненных окружениях при доставке приложений малых или средних размеров там, где необходимо сделать больше меньшими усилиями.

Архитектура Docker



Архитектура Docker

Демон Docker

Демон **Docker** работает на хост сервере.
Пользователь использует клиент **Docker** для взаимодействия с демоном.

Архитектура Docker

Клиент Docker

Клиент **Docker** в формате исполняемого файла `docker` — это основной пользовательский интерфейс для **Docker**. Он принимает команды и конфигурационные флаги от пользователя и взаимодействует с демоном **Docker**. Один клиент может взаимодействовать с множеством несвязанных демонов.

Архитектура Docker

Внутри Docker

Для понимания внутренностей Docker необходимо знать про

- образы (**images**),
- реестры (**registries**),
- контейнеры (**containers**).

Архитектура Docker

Образы Docker

Образ **Docker** — это шаблон в формате «только для чтения» с инструкциями для создания контейнера **Docker**. Например, образ может содержать в себе ОС Ubuntu с web-сервером Apache(Nginx) и установленным внутри web-приложением.

Можно собрать или обновить образ с нуля, или загрузить и использовать образы, созданные другими людьми.

Образ может быть основан или расширять один или более других образов.

Образ **Docker** описан в текстовом файле Dockerfile, который имеет простой и вполне определенный синтаксис.

Образы Docker — это основной строительный компонент Docker.

Архитектура Docker

Контейнеры Docker

Контейнер **Docker** — это запускаемый экземпляр **Docker** образа.

Можно запустить, остановить, переместить или удалить контейнер, используя Docker API или консольный клиент.

Каждый контейнер — это изолированная и безопасная платформа для приложений, однако, ему может быть предоставлен доступ к ресурсам, работающим на другом хосте или контейнере, например база данных.

Контейнеры **Docker** — это рабочий компонент **Docker**.

Архитектура Docker

Реестры Docker

Реестр **Docker** — это библиотека образов. Реестр может быть публичным, приватным, а также может быть установлен на том же сервере, что и демон или клиент **Docker** или же на совсем частном сервере.

Реестры Docker — это компонент распространения **Docker**.

Архитектура Docker

Сервис Docker

Сервис **Docker** позволяет целому рою (swarm) **Docker** узлов работать одновременно, обслуживая определенное количество экземпляров реплицированной задачи (replica task), представляющей собой образ **Docker**.

Возможно определить количество одновременных запускаемых реплицированных задач и кластерный менеджер убедится, что нагрузка распределена равномерно по рабочим узлам.

Для конечного потребителя сервис **Docker** представляется как одно приложение. Docker Engine поддерживает режим работы swarm mode с Docker 1.12 и выше.

Сервисы Docker — это компоненты масштабирования **Docker**.

Docker Hub

Docker Hub — это облачный репозиторий, предоставляемый Docker. Там пользователи могут создавать, тестировать, хранить и распространять образы контейнеров.

При помощи **Docker Hub** пользователь может получить доступ к публичным репозиториям образов с открытым кодом. Также он может использовать пространство хаба для создания собственных частных репозиторий, функций автоматизированной сборки, веб-хуков и т. п.

<https://docs.docker.com/>

СПАСИБО ЗА ВНИМАНИЕ