Лекция 12. Контейнеризация, Docker.

12.1. Что такое контейнеризация

Контейнеризация — это подход к разработке программного обеспечения, при котором приложение или служба, их зависимости и конфигурация упаковываются вместе в образ контейнера. Контейнерное приложение может тестироваться как единое целое и развертываться как экземпляр образа контейнера в операционной системе узла (сервера или виртуальной машины).

Так же как обычные контейнеры позволяют перевозить любые грузы на корабле, поезде или грузовике, программные контейнеры выступают в качестве стандартных модулей для развертывания программного обеспечения, которые могут содержать различный код и зависимости. Контейнеризация программного обеспечения позволяет разработчикам и ИТ-специалистам развертывать его в разных средах без каких-либо изменений или с минимальными изменениями.

Преимущества контейнеризации:

- Контейнеры изолируют приложения друг от друга в общей операционной системе;

Контейнерные приложения выполняются на основе узла контейнеров, который в свою очередь работает в операционной системе (Linux или Windows). Поэтому контейнеры требуют гораздо меньше ресурсов, чем образы виртуальных машин.

- Масштабируемость.

Вы можете быстро осуществлять горизонтальное масштабирование, создавая контейнеры для краткосрочных задач. С точки зрения приложения, создание экземпляра образа (контейнера) аналогично созданию экземпляра процесса, например для службы или веб-приложения. Но для обеспечения надежности при запуске нескольких экземпляров одного образа на нескольких серверах обычно желательно, чтобы контейнеры (экземпляры образа) выполнялись на разных серверах или виртуальных машинах в разных доменах сбоя.

Иными словами, контейнеры предоставляют такие преимущества, такие как переносимость, гибкость и контроль, на протяжении всего жизненного цикла приложения. Самым важным преимуществом является изоляция среды разработки от рабочей среды.

12.2. Что такое Docker?

Docker — это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде. Одновременно с этим, Docker — это компания, которая разрабатывает и продвигает эту технологию в сотрудничестве с поставщиками облачных служб, а также решений Linux и Windows, включая корпорацию Microsoft.

Контейнеры образов Docker работают в исходном формате в Linux и Windows. Но образы Windows будут выполняться только на узлах Windows, тогда как образы Linux —

на узлах Linux или Windows (на данный момент с помощью виртуальной машины Linux Hyper-V). Термин "узлы" здесь означает физические серверы и виртуальные машины.

Разработчики могут использовать среды разработки на базе Windows, Linux или macOS. На компьютере разработчика выполняется узел Docker, где развернуты образы Docker с создаваемым приложением и всеми его зависимостями. Разработчики, использующие Linux или Mac, могут применять узел Docker на базе Linux и создавать образы только для контейнеров Linux. В Windows разработчики могут создавать образы для контейнеров Linux или Windows.

Docker предоставляет версию Docker Community Edition (CE) для Windows или macOS, которая позволяет размещать контейнеры в среде разработки и предоставляет дополнительные средства разработки. Оба продукта устанавливают необходимую виртуальную машину (узел Docker) для размещения контейнеров. Docker также предлагает версию Docker Enterprise Edition (EE), которая предназначена для корпоративных разработчиков и ИТ-отделов, которые создают, распространяют и выполняют крупные и критически важные приложения в рабочей среде.

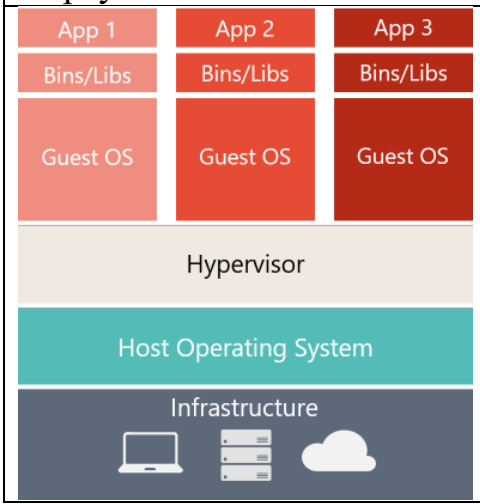
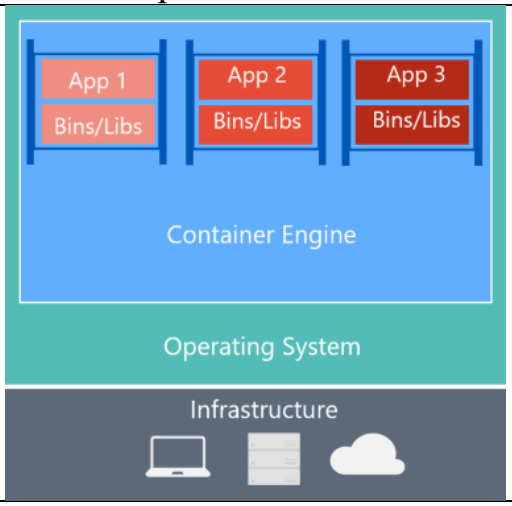
Для выполнения контейнеров Windows есть среды выполнения двух типов:

- Контейнеры Windows Server изолируют приложение с помощью технологии изоляции процесса и пространства имен. Контейнер Windows Server использует ядро совместно с узлом контейнеров и всеми остальными контейнерами на узле;
- Контейнеры Hyper-V увеличивают изоляцию, обеспеченную контейнерами Windows Server, запуская каждый контейнер в оптимизированной виртуальной машине. В этой конфигурации ядро узла контейнера не используется совместно с контейнерами Hyper-V, что улучшает изоляцию.

Образы для этих контейнеров создаются и работают одинаково. Различие заключается лишь в том, что для создания контейнера из образа с контейнером Hyper-V нужен дополнительный параметр.

12.3. Сравнение контейнеров Docker с виртуальными машинами

На рисунке ниже показано сравнение между виртуальными машинами и контейнерами Docker.

Виртуальная машина	Контейнер Docker
	

Виртуальные машины содержат приложение, необходимые библиотеки или двоичные файлы и всю операционную систему. Полная виртуализация требует больше ресурсов, чем создание контейнеров.	Контейнеры включают в себя приложение и все его зависимости. Но они используют ядро ОС совместно с другими контейнерами, которые выполняются в изолированных процессах в пользовательском пространстве операционной системы узла. (Это не относится к контейнерам Hyper-V, где каждый контейнер запускается на отдельной виртуальной машине.)

Так как контейнеры требуют гораздо меньше ресурсов (например, им не нужна полная ОС), их проще развертывать и они быстрее запускаются. Это позволяет повысить плотность развертываний, то есть запустить на одной единице оборудования больше служб и сократить затраты на них.

Запуск на одном ядре приводит к тому, что уровень изоляции будет ниже, чем на виртуальных машинах.

Основная цель образа — привести среду (зависимости) к единообразию в различных развертываниях. Это означает, что вы можете отладить образ на одном компьютере, а затем развернуть его на другом компьютере и получить ту же среду.

Образ контейнера — это способ упаковки приложения или службы для надежного и воспроизводимого развертывания. Можно сказать, что Docker является не только технологией, но еще философией и процессом.

При работе с Docker разработчики никогда не жалуются, что приложение работает только на локальном компьютере, но не в рабочей среде. Им достаточно сказать "Выполняется в Docker", так как упакованное приложение Docker будет выполняться в любой поддерживаемой среде Docker. Оно будет работать одинаково во всех сценариях развертывания (разработка, контроль качества, промежуточное размещение и рабочая среда).

Docker каждый слой представляет некоторый набор изменений, которые применяются к файловой системе после выполнения команды, такой как установка программы.

Если вы "посмотрите" на файловую систему после копирования очередного слоя, вы увидите все файлы в том состоянии, которое они приняли после установки программы.

Такой образ можно рассматривать как дополнительный жесткий диск, доступный только для чтения, который готов к установке на "компьютер" с уже установленной операционной системой.

Соответственно, роль "компьютера" здесь выполняет контейнер, в который устанавливается жесткий диск этого образа. Контейнер, как и обычный компьютер, можно включать и отключать.

12.4 Терминология Docker

В этом разделе приводятся термины и определения, необходимые при работе с инструментом Docker.

Образ контейнера — пакет со всеми зависимостями и сведениями, необходимыми для создания контейнера. Образ включает в себя все зависимости (например, платформы), а также конфигурацию развертывания и выполнения для среды выполнения контейнера. Как правило, образ создается на основе нескольких базовых образов, наслоенных друг на друга в файловой системе контейнера. После создания образ остается неизменным.

Dockerfile — текстовый файл, содержащий инструкции по сборке образа Docker. Он похож на пакетный сценарий, где первая строка указывает базовый образ, с которого начинается работа, а следующие инструкции устанавливают необходимые программы, копируют файлы и т. п. для создания необходимой рабочей среды.

Сборка — действие по созданию образа контейнера на основе сведений и контекста, предоставленных файлом Dockerfile, а также дополнительных файлов в папке, где создается образ. Сборка образа выполняется с помощью команды **docker build**.

Контейнер — экземпляр образа Docker. Контейнер отвечает за выполнение одного приложения, процесса или службы. Он состоит из содержимого образа Docker, среды выполнения и стандартного набора инструкций. При масштабировании службы вы создаете несколько экземпляров контейнера из одного образа. Или пакетное задание может создать несколько контейнеров из одного образа, передавая разные параметры каждому экземпляру.

Реестр — служба, предоставляющая доступ к репозиториям. Реестр по умолчанию для большинства общедоступных образов — Центр Docker (принадлежащий Docker как организации). Реестр обычно содержит репозитории нескольких команд. Компании часто используют частные реестры для хранения своих образов и управления ими. Еще один пример — реестр контейнеров Azure.

Кластер — коллекция узлов Docker, представленная в виде единого виртуального узла Docker, чтобы можно было масштабировать приложение в нескольких экземплярах служб, распределенных по нескольким узлам кластера. Кластеры Docker можно создавать с помощью Kubernetes, Azure Service Fabric, Docker Swarm и (или) Mesosphere DC/OS.

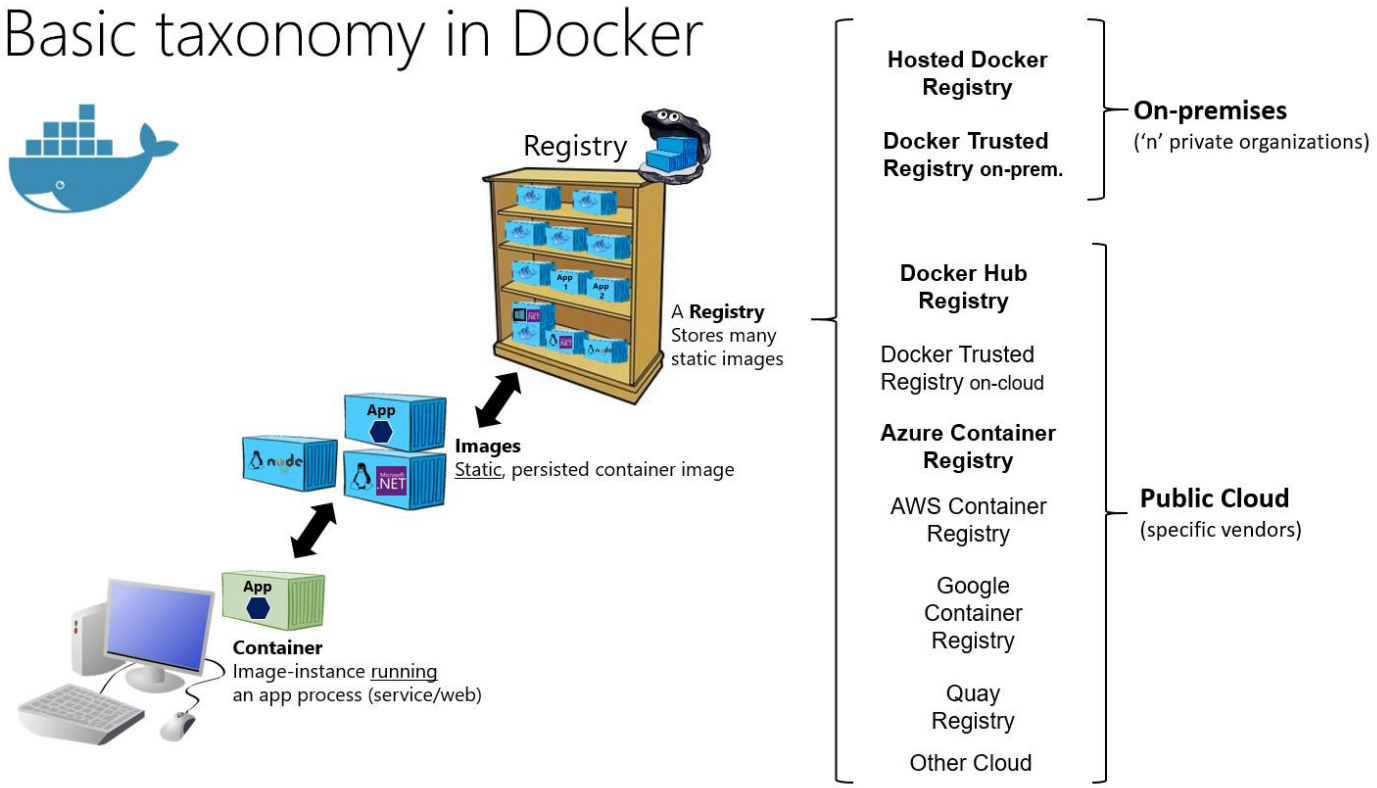
12.5 Контейнеры, образы и реестры Docker

При использовании Docker разработчик создает приложение или службу и упаковывает приложение или службу и их зависимости в образ контейнера. Образ — это статическое представление приложения или службы, а также их конфигурации и зависимостей.

Для запуска приложения или службы создается экземпляр образа приложения, чтобы создать контейнер, который будет запущен на узле Docker. Контейнеры изначально проверяются в среде разработки или на ПК.

Разработчикам следует хранить образы в реестре, который выступает в качестве библиотеки образов и необходим при развертывании на оркестраторы в рабочей среде. Docker поддерживает общедоступный реестр с помощью Docker Hub. Другие поставщики предлагают реестры для различных коллекций образов, включая Реестр контейнеров Azure. Кроме того, организации могут развернуть локальные частные реестры для своих образов Docker.

На рисунке ниже показано, как образы и реестры в Docker связаны с другими компонентами. На нем также показано несколько вариантов реестра от поставщиков.



Классификация основных понятий и терминов Docker

Размещение образов в реестре позволяет хранить статические и неизменяемые фрагменты приложений, включая всех их зависимости на уровне платформы. Эти образы можно добавить в систему управления версиями и развернуть в нескольких средах, таким образом, каждый образ представляет согласованную единицу развертывания.

Частные реестры образов, размещенные локально или в облаке, рекомендуется использовать, если:

- Ваши образы не могут быть открыты для общего доступа по соображениям конфиденциальности;
- Вам необходимо обеспечить минимальную сетевую задержку между образами и выбранной средой развертывания. Например, если рабочая среда представляет собой облако Azure, вы, вероятно, захотите разместить образы в Реестре контейнеров Azure, чтобы сетевая задержка была минимальной. Точно так же, если рабочая среда является локальной, вы можете развернуть локальный доверенный реестр Docker в той же локальной сети.

Рабочий процесс разработки для приложений Docker

Жизненный цикл разработки приложений начинается на компьютере каждого разработчика, где разработчик локально программирует приложение на предпочитаемом языке и тестирует его. Независимо от выбранного языка, инфраструктуры и платформы,

в рамках этого рабочего процесса разработчик всегда разрабатывает и тестирует контейнеры Docker, но делает это локально.

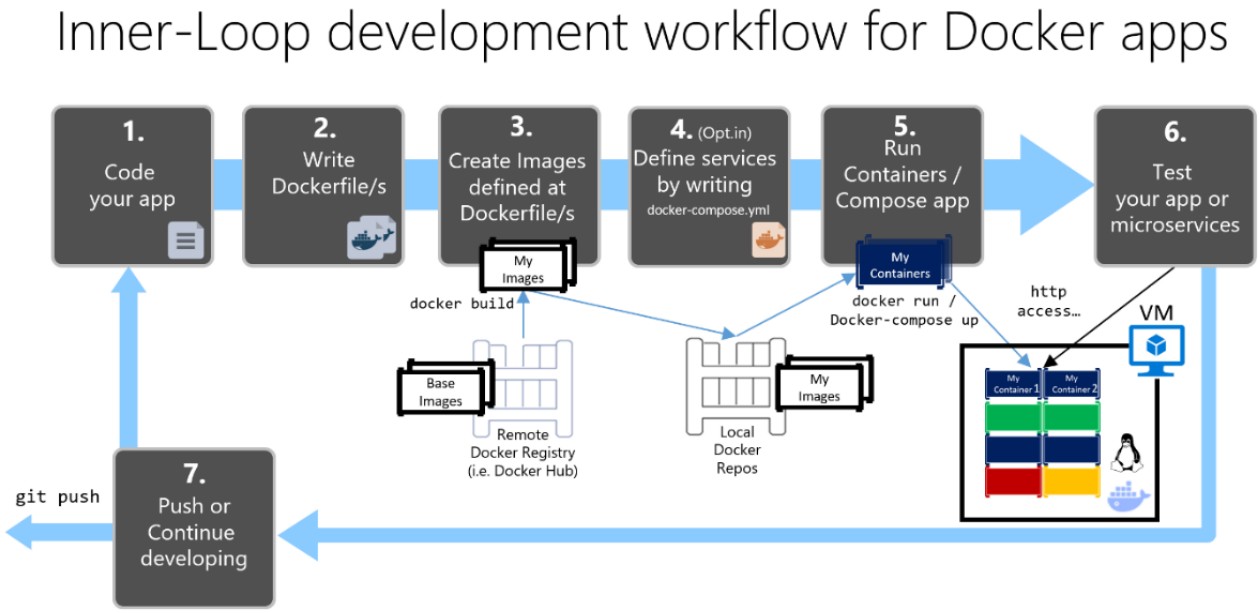
В каждый контейнер (экземпляр образа Docker) входят следующие компоненты:

- Выбранная операционная система (например, дистрибутив Linux, Windows Nano Server или Windows Server Core);
- Файлы, добавленные разработчиком (двоичные файлы приложения и т. п.);
- Сведения о конфигурации (параметры среды и зависимости).

Рабочий процесс разработки приложений Docker на основе контейнера

В этом разделе описывается рабочий процесс *внутреннего цикла* разработки приложений на основе контейнера Docker. Рабочий процесс внутреннего цикла означает, что речь идет только о разработке, которая выполняется на компьютере разработчика, не касаясь более широкого рабочего процесса DevOps. Начальные этапы настройки среды здесь не рассматриваются, так как они выполняются только один раз.

Приложение состоит из ваших собственных служб и дополнительных библиотек (зависимостей). Ниже приведены основные шаги, которые обычно выполняются при сборке приложения Docker, как показано на рисунке ниже.



Пошаговый рабочий процесс разработки приложения на основе контейнера Docker

При использовании Visual Studio многие из этих шагов выполняются автоматически, что значительно повышает производительность. Это особенно справедливо в тех случаях, когда используется Visual Studio 2017 и планируется создание много контейнерных приложений. Например, всего лишь одним щелчком мыши Visual Studio добавляет в проект *Dockerfile* и файл *docker-compose.yml* с конфигурацией для вашего приложения. При запуске приложения в Visual Studio он создает образ Docker и запускает много контейнерное приложение непосредственно в Docker. Вы даже можете отлаживать несколько контейнеров одновременно. Эти возможности значительно повышают скорость разработки.

Dockerfile — скрипт, который позволяет автоматизировать процесс построения контейнеров — шаг за шагом, используя при этом **base** образ.

Использованные источники:

- https://docs.microsoft.com/ru-ru/dotnet/standard/microservices-architecture/container-docker-introduction/
- https://ru.wikipedia.org/wiki/Docker