УДК 004.4

КОНТЕЙНЕРИЗАЦИЯ И КАК В ЭТО ПОГРУЗИТЬСЯ

Киуло А. Ю.
Научный руководитель – доцент, к.т.н. Макареня С. Н.

В данной научной работе мы рассмотрим основные понятия, позволяющие раскрыть тему контейнеризации, рассмотрим, чем контейнеризация отличается от виртуализации, так как эти понятия очень схожи, и нам необходимо провести тонкую грань, разграничивающую их. Также мы рассмотрим архитектуру самого популярного средства контейнеризации Docker, расскажем о достоинствах и недостатках технологии в целом.

В современном мире существуют десятки различных операционных систем и аппаратных компонентов. И с каждым годом становится все сложнее и сложнее поддерживать один и тот же программный продукт для различных сред выполнения, многие из программ вообще никогда не портировались под некоторые из сред.

С течением времени и развитием аппаратного обеспечения вычислительных машин закономерным шагом в развитии программного обеспечения и, в частности, операционных систем стало создание такого механизма как виртуализация.

Виртуализация – это запуск одной или нескольких систем в рамках другой операционной системы на одной физической машине. Она подразумевает набор логически изолированных вычислительных ресурсов в виде абстракций над аппаратной реализацией.

Частным случаем виртуализации стала контейнеризация. Благодаря механизму изоляции процессов (namespace) и контрольных групп (cgroups) стала возможна изоляция процессов в контейнерах. Процесс, запущенный в контейнере, выполняется внутри операционной системы хоста, но при этом он изолирован от остальных процессов. Для самого процесса это выглядит так, будто он единственный работает в системе.

Для более легкого понимания введем основные понятия, которые помогут раскрыть тему в полной мере. К основным понятиям в контейнеризации можно отнести: container image, registry server, container, container engine и container runtime.

Начнем с container image. Container image (образ) – представляет собой файл, в который упаковывается как приложение, так и его среда. Образ также содержит систему файлов, которая станет доступна приложению, в добавок образ будет включать в себя метаданные (к примеру команды, выполняющиеся при запуске контейнера). Принято считать, что container image состоят из слоев (по конвенции один слой, должен содержать в себе одну инструкцию). Различные образы имеют все шансы содержать одинаковые слои, так как каждая прослойка надстроена поверх иного образа, а два разных образа смогут брать за основу и использовать одинаковые родительские образы. Container images сохраняются в Registry Server (реестре) и с помощью tag (тегов) к ним присваиваются версии. В случае если тег не указан в процессе разработки, то по умолчанию применяется latest тег.

Следующим понятием, которое необходимо рассмотреть, является Registry Server. Registry Server – это хранилище, в котором сохранятся образы. После создания образа на локальном компьютере его можно отправить (push) в хранилище, а затем извлечь (pull) на другом компьютере и запустить его там.

Container (контейнер) – это копия образа контейнера. Выполняемый контейнер – это процесс, отделенный от остальных процессов на сервере и лимитированный назначенным объемом ресурсов (ЦПУ, ОЗУ, диска и др.). Выполняемый контейнер хранит все без исключения слои образа с доступом на чтение и в дополнение создает собственный выполняемый слой с доступом на запись. Данный создаваемый слой и отличает контейнер от образа.

Следующим рассматриваемым понятием станет движок контейнеризации. Container Engine – это программный элемент, который управляет контейнерами и образами. Движок контейнеризации дает возможность формировать, публиковать, а также скачивать образы.

Последним, но не менее важным рассматриваемым определением будет среда выполнения контейнеров. Container Runtime – это программный элемент, позволяющий создавать и запускать контейнеры.

Сегодня я хочу остановиться и рассказать подробнее про Docker. Он состоит из утилиты командной строки, которая вызывает одноименный сервис.

В настоящее время Docker стал стандартом фактически для многих ИТ-администраторов и занимает значительную долю разработчиков. Docker предоставляет функции для управления контейнерами, но также в нем присутствуют уязвимости системы безопасности.

Наиболее распространенным Container Engine на сегодняшний день является Docker.

Что такое Docker Engine?

Docker Engine клиент-серверное приложение с тремя главными компонентами:

1. Сервер работающий в фоновом режиме (демон).
2. REST API, который используют программы для взаимодействия с сервером.
3. Интерфейс командной строки (CLI) клиент.

Рассмотрим, как происходят процессы в Docker при использовании клиент-серверной архитектуры. Docker клиент обращается к демону, который создает, запускает и доставляет ваши контейнеры. CLI (Command Line Interface) отправляет команды демону, а затем демон воздействует на данные команды для создания контейнеров. Docker клиент и демон могут быть запущены в одной системе или клиент может подключиться к удаленному демону. Docker клиент и демон общаются через сокеты или REST API.

Docker также позволяет контролировать и настраивать помимо окружения сетевые интерфейсы, монтировать на чтение или чтение и запись часть файловой системы хоста, а также ограничивать системные ресурсы, такие как процессорное время и объем занимаемой оперативной памяти, но это уже выходит за рамки нашего вводного знакомства.

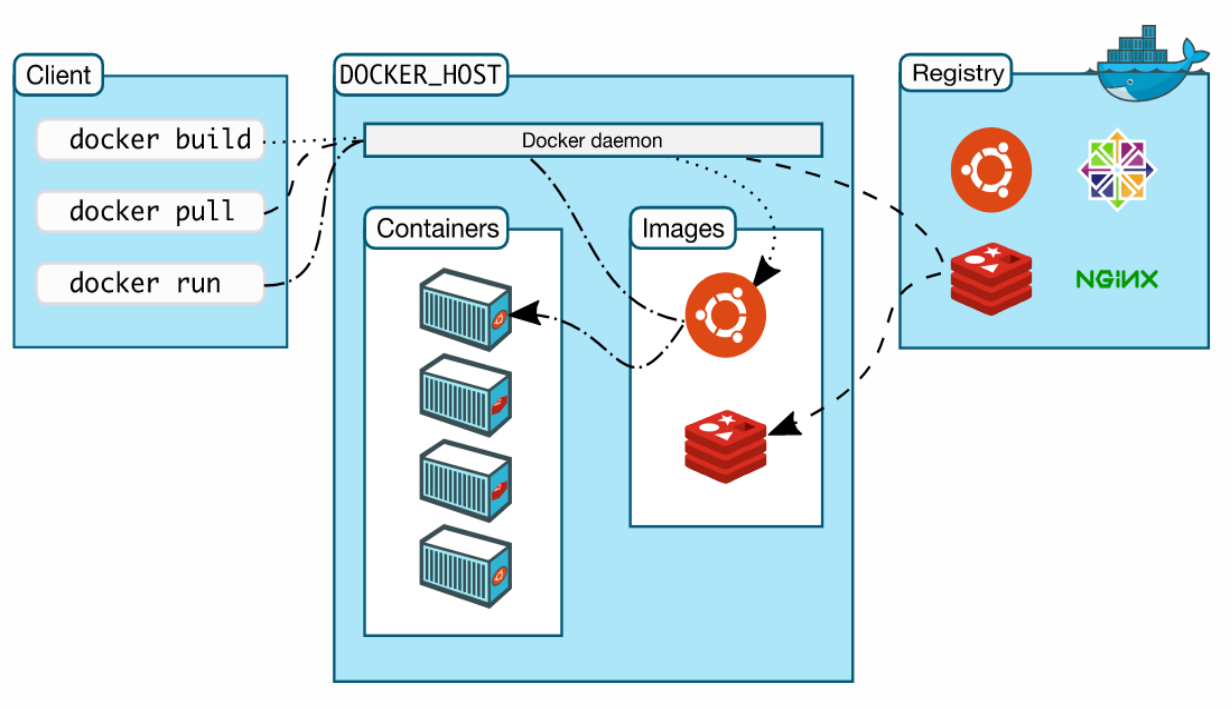

Рис. 1 – Архитектура Docker

Для установки необходимо воспользоваться официальным руководством – которое содержит подробные инструкции для Linux, Windows и Mac. Стоит сразу отметить, что контейнерам для работы необходимы функции ядра Linux, поэтому они работают нативно под Linux, почти нативно в последних версиях Windows благодаря WSL2 (через Docker Desktop или Linux диструбутив) и не нативно под Mac (используется виртуализация).

Работа с Docker начинается с описания образа. Для этого принято использовать Dockerfile.

Далее для создания образа надо вызвать утилиту docker с командой create указав путь до каталога контекста, по умолчанию docker будет искать Dockerfile в этом каталоге.

После этого мы увидим, что у нас появился новый образ с помощью команды docker images.

Далее мы должны создать и запустить контейнер – он же экземпляр нашей программы с помощью команды docker run <image name>.

Список запущенных контейнеров можно просмотреть с помощью команды docker ps.

К достоинствам контейнеризации можно отнести:

1. Скорость создания контейнера. Контейнеризация дает возможность создать контейнер быстрее, чем ВМ. Наряду с этим среда контейнеризации для определенных проблем дает больше преимуществ.

2. Экономичность. Контейнер использует меньше памяти в хранилище, что уменьшает дополнительные траты.

3. Высокая производительность. Неимение межсетевых связей и коллизий улучшает эффективность разработки. Любой контейнер на практике является микросервисом, который можно автономно развивать, не задаваясь вопросом синхронизации.

4. Управление версиями. Есть возможность отслеживать версии контейнеров, контролировать различия между контейнерами.

5. Потенциальная возможность переноса среды вычислений. Все отношения приложений и ОС, требуемые для функционирования приложения, инкапсулируются. Это дает возможность без значительных трудозатрат интегрировать образ контейнера в различных средах.

6. Стандартизация. Чаще всего, контейнеры строятся на базе открытых стандартов. Данная возможность позволяет работать с ними в большинстве дистрибутивов Linux, Microsoft, MacOS.

7. Безопасность. Контейнеры отделены между собой и также отделены от базовой инфраструктуры. Любые манипуляции, такие как изменение, обновление или удаление одного контейнера не будет влиять на другой контейнер.

Как и у любой другой технологии, у Docker есть свои недостатки:

1. Высокая сложность. Увеличение числа контейнеров, функционирующих в рамках одного приложения, влияет на сложность администрирования данных контейнеров.

2. Разрастание. Довольно часто в контейнеры добавляется значительно больше ресурсов, чем действительно этого требует приложение. Из-за этого образ расширяется, занимая больше памяти на диске.

3. Поддержка Native Linux. Docker и большинство других контейнерных технологий основываются на Linux-контейнерах (LXC). В связи с этим запуск контейнеров в операционной системе Windows не всегда практичен, а постоянная эксплуатация сложнее, чем при работе в Linux.

Литература

1. Контейнеризация [Электронный ресурс]. – Информационный ресурс Wikimedia Foundation. URL: https://ru.wikipedia.org/wiki /%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F (дата обращения: 25.04.2022).

2. Что Такое Docker: Для Чего Он Нужен и Где Используется [Электронный ресурс]. – Информационный ресурс Блог Компании Селектел. URL: https://selectel.ru/blog/what-is-docker/ (дата обращения: 25.04.2022).