ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

https://doi.org/10.21122/1683-6065-2022-3-108-111
УДК 669.27.519

Поступила 21.06.2022
Received 21.06.2022

ПРИМЕНЕНИЕ ВИРТУАЛЬНЫХ КОНТЕЙНЕРОВ DOCKER ДЛЯ ЗАПУСКА СЕРВИСОВ

И. А. БУРЕНКОВ, ОАО «БМЗ – управляющая компания холдинга «БМК», г. Жлобин, Гомельская обл., Беларусь, ул. Промышленная, 37. E-mail: bia.asu@bmz.gomel.by, тел.: 80233455166

Docker – инструмент виртуализации, который позволяет разработчикам и администраторам систем запускать приложения и сервисы в изолированных друг от друга и от гостевой операционной системы контейнерах. Контейнеры Docker в отличие от прочих инструментов виртуализации работают напрямую с аппаратным обеспечением хостовой машины, что благоприятно сказывается на расходе ресурсов системы. Контейнеры запускаются из заранее собранных образов. Возможность собрать в образе все зависимости и данные приложения позволяет не беспокоиться о совместимости с системами, в которых будет запущен контейнер. Независимость контейнеров друг от друга позволяет запускать целые группы связанных сервисов, в которых изменение версии или данных в одном из контейнеров не оказывает влияния на функционирование остальной группы.

Ключевые слова. *Docker, виртуализация, контейнер, образ, сервис, развертывание, Linux.*
Для цитирования. *Буренков, И. А. Применение виртуальных контейнеров Docker для запуска сервисов / И. А. Буренков // Литье и металлургия. 2022. № 3. С. 108–111. https://doi.org/10.21122/1683-6065-2022-3-108-111.*

USING DOCKER VIRTUAL CONTAINERS TO LAUNCH SERVICES

I. A. BURENKOV, OJSC «BSW – Management Company of Holding «BMC», Zhlobin, Gomel region, Belarus, 37, Promyshlennaya str. E-mail: bia.asu@bmz.gomel.by; tel.: 80233455166

Docker is a virtualization tool that allows developers and system administrators to run applications and services in containers isolated from each other and from the guest operating system. Docker containers, unlike other virtualization tools, work directly with the host machine hardware, which is good for system resource consumption. Containers run from prebuilt images. The ability to collect all dependencies and application data in an image allows you not to worry about compatibility with the systems in which the container will run. The independence of containers from each other allows you to run entire groups of related services, in which a change of version or data in one of the containers does not affect the functioning of the rest of the group.

Keywords. *Docker, virtualization, container, image, service, deployment.*
For citation. *Burenkov I. A. Using Docker virtual containers to launch services. Foundry production and metallurgy, 2022, no. 3, pp. 108–111. https://doi.org/10.21122/1683-6065-2022-3-108-111.*

Docker – это open-source инструмент виртуализации, который позволяет разработчикам и администраторам систем запускать приложения в изолированных друг от друга и от гостевой операционной системы контейнерах. Принцип работы похож на классические виртуальные машины, однако имеет ряд отличий. Если гипервизоры типа KVM, HyperV и прочие создают виртуальное представление аппаратного обеспечения сервера, Docker использует ресурсы сервера напрямую, тем самым, обеспечивая более быстрый запуск и низкое потребление ресурсов системы (рис. 1).

Основной сферой применения Docker является ускоренное развертывание и тестирование приложений и сервисов. Контейнеры, не требующие зависимостей в системе, позволяют запускать разные версии программного обеспечения одновременно, не опасаясь конфликтов, что может быть полезно при разработке, обновлении или миграции проектов.

Преимущество Docker также состоит в отображении приложения в контейнере. Можно собрать в одном контейнере целое дерево процессов, например само приложение, веб-сервер, базу данных и т. д. Но наиболее эффективно будет разделить все основные процессы в разные контейнеры, тогда обновления или ошибки в одном из них не затронут остальные. Для связи контейнеров между собой и с сетевыми устройствами Docker использует виртуальные сети, может назначать ip адреса и пробрасывать порты. Каждый контейнер имеет свою собственную файловую систему, сетевой стек, пространство процессов.

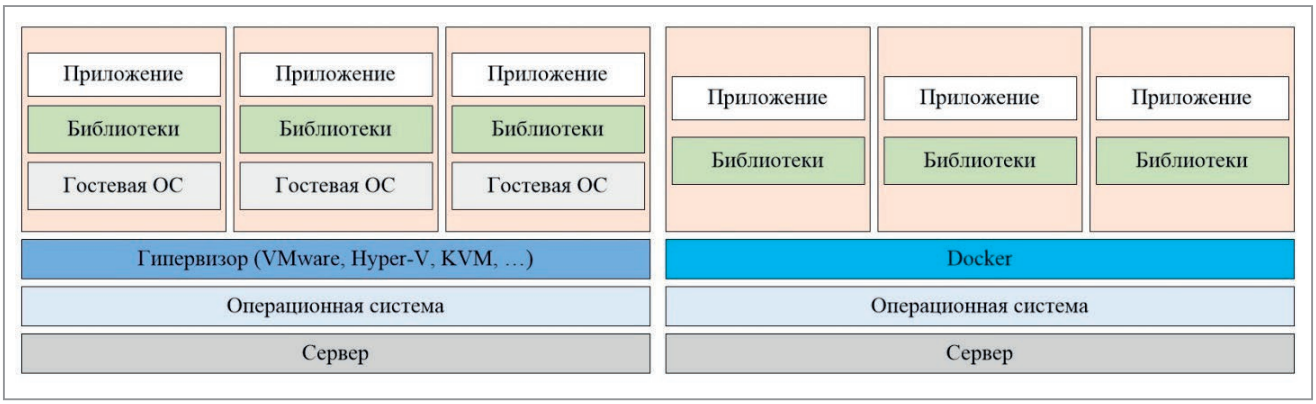

Рис. 1. Отличия между Docker и другими виртуализаторами

Контейнеры изолированы и от основной операционной системы и друг от друга, таким образом, внесение изменений в них не оказывает никакого влияния на систему и другие запущенные контейнеры, в том числе, не позволяя случайно или умышленно навредить основной системе. Один из важных плюсов – возможность в любой момент перезапустить, остановить или удалить контейнер. Если есть необходимость сохранять данные во время работы приложения, Docker позволяет сделать это несколькими способами. Первый способ представляет собой создание независимого от контейнера тома для хранения данных. По умолчанию том создается на хосте, но можно подключать и использовать удаленные хранилища и файловые сервера. Второй способ заключается в монтировании каталога хостовой операционной системы в контейнер. Способ имеет свои минусы, например, потеря данных при миграции приложения на другой сервер или использовании данных несколькими приложениями. Также для сохранения внесенных изменений существует команда “docker commit”. Она генерирует из выбранного контейнера новый образ, содержащий в себе все данные и выполненные действия во время работы. Каждый контейнер имеет свою собственную файловую систему, сетевой стек, пространство процессов.

Образ – файл, из которого создаются контейнеры. Он содержит в себе сам код, среду выполнения, драйверы, инструменты, библиотеки и т.д. Образы создаются при помощи специального файла (Dockerfile), подробно описывающего все действия, которые необходимо выполнить в процессе сборки. Туда могут входить операции по копированию файлов в образ, указания на открытие портов, назначение переменных окружения, выполнение команд непосредственно в оболочку базового образа и многое другое.

Одно из важнейших преимуществ Docker состоит в особенностях сборки образа. Так, при выполнении каждой команды из Dockerfile происходит как бы наложение слоя на базовый образ, который можно представить как набор файлов, созданных или измененных командой. Это позволяет улучшить производительность системы за счет кэширования слоев, которые не меняются от сборки к сборке. Кэшированные слои также используются при создании других образов, если они идентичны, что позволяет ускорить процесс и экономит место на диске.

Наличие образов также облегчает развертывание на множество устройств. Разработчику не придется повторять множество действий по настройке и установке на каждой машине. Достаточно один раз собрать образ со всем необходимым. Существование плагинов таких, как “docker-compose”, расширяют возможности приложения, позволяя разворачивать не один, а целое множество сервисов одной командой.

Для хранения и обмена образами существует официальный публичный репозиторий под названием Docker Hub, в который можно как загружать свои, так и скачивать образы, созданные другими пользователями, в том числе официальные образы от компаний-разработчиков и поддерживаемые сообществом Docker (Ubuntu, Java, Nginx, Python и т.д.). Для нужд компаний также имеется возможность создать приватный репозиторий. При скачивании готовых образов из репозитория имеется возможность выбрать версию образа, обозначенную разработчиком, с помощью указания тегов (например, nginx: latest, nginx:1.20.2, nginx:1.21-perl).

Рассмотрим несколько проектов Docker, использующихся в производстве.

Проект audit представляет собой сервис массовых рассылок уведомлений о предстоящих аудитах. В основе контейнера лежит образ ОС Debian 11 с интегрированной программной платформой Node.js, на языке которой написано web-приложение. Для полноценной работы сервиса необходимо добавить в основной образ зависимости и скопировать файлы приложения. Ниже приведен листинг файла, описывающий процесс сборки:

```
FROM node:16
RUN npm cache clean –force
RUN npm install -g nodemon
WORKDIR /home/node/app
COPY. /home/node/app/
RUN npm install
RUN chmod +x entrypoint.sh
RUN chown -R node: node /home/node/app
USER node
EXPOSE 8080
ENTRYPOINT /home/node/app/entrypoint.sh
```

С помощью описанных в Dockerfile инструкций был собран новый образ сервиса. Во время запуска Docker пробрасывает порт контейнера на хостовую машину, где с помощью веб-сервера Apache организовывается доступ к сервису.

Второй проект используется для переноса и тестирования обновления баз данных системы мониторинга Zabbix. Он состоит из нескольких контейнеров: сервер Zabbix, веб-фронтенд Zabbix, PhPMyAdmin и база данных MySQL (рис. 2). Организовано подключение файлов конфигурации и данных в виде примонтированного каталога, что исключает сброс настроек при перезапуске контейнера или смене версии приложения внутри него.

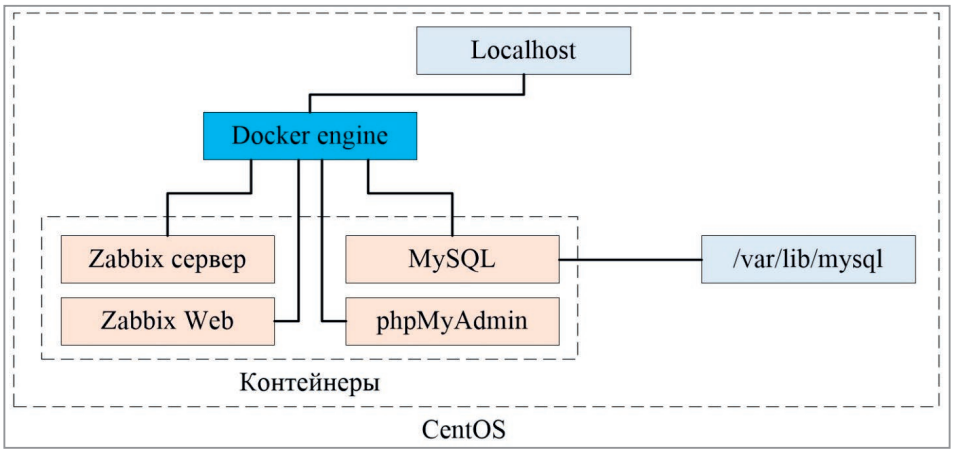

Рис. 2. Проект для тестирования обновлений Zabbix

Запуск группы контейнеров организован через плагин “docker-compose”. Конфигурационный файл, написанный на языке YAML, указывает все условия и действия, необходимые для функционирования группы:

```
services:
  zabbix:
    image: zabbix/zabbix-server-mysql: latest
    container_name: zabbix-server
    hostname: zabbix-server
    network_mode: host
    env_file:
    -  zabbix.env
  ports:
    - 10051:10051
    - 10050:10050
  restart: always
  mysql:
  image: mysql
  container_name: mysql
```

```
hostname: mysql
network_mode: host
env_file:
   -  mysql.env
volumes:
   - /opt/zabbix-web-mysql/mysql-data:/var/lib/mysql
   security_opt:
   -  seccomp: unconfined
ports:
   - 3306:3306
   - 33060:33060
restart: always
zabbix-web:
image: zabbix/zabbix-web-apache-mysql
container_name: zabbix-web
hostname: zabbix-web
network_mode: host
env_file:
   -  web.env
ports:
   - 8080:8080
   - 8443:8443
restart: always
```

В совокупности это позволяет получить ряд преимуществ по сравнению с классическим развертыванием. Так, смена версии программного обеспечения в одном из контейнеров не повлияет на работу других и позволит провести проверки совместимости и миграцию баз данных.

Таким образом, можно отметить, что Docker является мощным и гибким инструментом для разработчиков и администраторов систем, позволяющим упростить, ускорить и автоматизировать развертывание приложений и сервисов. Контейнеры позволяют разработчикам не думать об инфраструктуре, в которой будет запущено их приложение, нужно ли вносить изменения в настройки или устанавливать зависимости.