ОСНОВЫ РАБОТЫ С DOCKER

Введение

Docker — это программное обеспечение с открытым исходным кодом, применяемое для разработки, тестирования, доставки и запуска веб-приложений в средах с поддержкой контейнеризации.

Он нужен для более эффективного использования системы и ресурсов, быстрого развертывания готовых программных продуктов, а также для их масштабирования и переноса в другие среды с гарантированным сохранением стабильной работы.

Введение

Разработка Docker была начата в 2008 году, а в 2013 году он был опубликован как свободно распространяемое ПО под лицензией Apache 2.0. В качестве тестового приложения Docker был включен в дистрибутив Red Hat Enterprise Linux 6.5. В 2017 году была выпущена коммерческая версия Docker с расширенными возможностями.

Docker работает в Linux, ядро которого поддерживает cgroups, а также изоляцию пространства имен.

Основной принцип работы Docker — контейнеризация приложений. Этот тип виртуализации позволяет упаковывать программное обеспечение по изолированным средам — контейнерам. Каждый из этих виртуальных блоков содержит все нужные элементы для работы приложения. Это дает возможность одновременного запуска большого количества контейнеров на одном хосте.

Введение

Преимущества **Docker**:

- Минимальное потребление ресурсов — контейнеры не виртуализируют всю операционную систему, а используют ядро хоста и изолируют программу на уровне процесса. Последний потребляет намного меньше ресурсов локального компьютера, чем виртуальная машина.
- Скоростное развертывание — вспомогательные компоненты можно не устанавливать, а использовать уже готовые **docker-образы** (шаблоны). Например, не имеет смысла постоянно устанавливать и настраивать **Linux Ubuntu**. Достаточно 1 раз ее инсталлировать, создать образ и постоянно использовать, лишь обновляя версию при необходимости.
- Удобное скрытие процессов — для каждого контейнера можно использовать разные методы обработки данных, скрывая фоновые процессы.

Введение

- Работа с небезопасным кодом — технология изоляции контейнеров позволяет запускать любой код без вреда для ОС.
- Простое масштабирование — любой проект можно расширить, внедрив новые контейнеры.
- Удобный запуск — приложение, находящееся внутри контейнера, можно запустить на любом docker-хосте.
- Оптимизация файловой системы — образ состоит из слоев, которые позволяют очень эффективно использовать файловую систему.

Определения

1. Docker-демон (Docker-daemon) — сервер контейнеров, входящий в состав программных средств Docker. Демон управляет Docker-объектами (сети, хранилища, образы и контейнеры). Демон также может связываться с другими демонами для управления сервисами Docker.
2. Docker-клиент (Docker-client / CLI) — интерфейс взаимодействия пользователя с Docker-демоном. Клиент и Демон — важнейшие компоненты «движка» Докера (Docker Engine). Клиент Docker может взаимодействовать с несколькими демонами.
3. Docker-образ (Docker-image) — файл, включающий зависимости, сведения, конфигурацию для дальнейшего развертывания и инициализации контейнера.
4. Docker-файл (Docker-file) — описание правил по сборке образа, в котором первая строка указывает на базовый образ. Последующие команды выполняют копирование файлов и установку программ для создания определенной среды для разработки.
5. Docker-контейнер (Docker-container) — это легкий, автономный исполняемый пакет программного обеспечения, который включает в себя все необходимое для запуска приложения: код, среду выполнения, системные инструменты, системные библиотеки и настройки.

Определения

6. **Том (Volume)** — эмуляция файловой системы для осуществления операций чтения и записи. Она создается автоматически с контейнером, поскольку некоторые приложения осуществляют сохранение данных.

7. **Реестр (Docker-registry)** — зарезервированный сервер, используемый для хранения d**ocker-образов**.

Примеры реестров:

- Центр **Docker** — реестр, используемый для загрузки **docker-image**. Он обеспечивает их размещение и интеграцию с **GitHub** и **Bitbucket**.
- Контейнеры **Azure** — предназначен для работы с образами и их компонентами в директории **Azure (Azure Active Directory)**.
- Доверенный реестр **Docker** или **DTR** — служба **docker-реестра** для инсталляции на локальном компьютере или сети компании.

Определения

8. Docker-хаб (Docker-hub) или хранилище данных — репозиторий, предназначенный для хранения образов с различным программным обеспечением. Наличие готовых элементов влияет на скорость разработки.

9. Docker-хост (Docker-host) — машинная среда для запуска контейнеров с программным обеспечением.

10. Docker-сети (Docker-networks) — применяются для организации сетевого интерфейса между приложениями, развернутыми в контейнерах.

11. Docker Engine («Движок» Docker) — ядро механизма Докера. «Движок» отвечает за функционирование и обеспечение связи между основными Docker-объектами (реестром, образами и контейнерами.

Введение

Быстрая доставка приложений (команды **docker pull** и **docker push**) позволяет организовать коллективную работу над проектом. Разработчики могут работать удаленно на локальных компьютерах и выполнять пересылку фрагментов кода в контейнер для тестов.

Развертывание и масштабирование — контейнеры работоспособны на локальных компьютерах, серверах, в облачных онлайн-сервисах. Их можно загружать на хостинг для дальнейшего тестирования, создавать (**docker run**), останавливать (**docker stop**), запускать (**docker start**), приостанавливать и возобновлять (**docker pause** и **docker unpause** соответственно).

Введение

Множественные нагрузки — осуществление запуска большого количества контейнеров на одном и том же оборудовании, поскольку Docker занимает небольшой объем дисковой памяти.

Диспетчер процессов — возможность мониторинга процессов в Docker посредством команд **docker ps** и **docker top**, имеющими схожий синтаксис с Linux.

Удобный поиск — в реестрах Docker он осуществляется очень просто. Для этого следует использовать команду **docker search**.

Установка Docker

Для установки Docker необходимо настроить репозиторий:

apt update

Установить дополнительные пакеты:

apt install ca-certificates curl gnupg lsb-release

Добавьте официальный GPG-ключ Docker:

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg

--dearmor -o /etc/apt/keyrings/docker.gpg

Настройка репозитория

Настройка репозитория Docker:

```
echo \ "deb [arch=$(dpkg --print-architecture) signed by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Установка Engine

Установка Docker Engine:

apt update

sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin

Hello-World образ

После установки необходимо убедиться, что установка Docker Engine прошла успешно, запустив hello-world образ:

```
docker run hello-world
```

Hello-World образ

```
root@inexpro-vps3:~# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

root@inexpro-vps3:~#
```

Создание образа Docker

Развертывать образ можно любое количество раз на любом хосте. Для создания образа используется один из двух способов: интерактивный или через **Dockerfile**.

Интерактивный — простой способ, при котором разработчик сам изменяет среду окружения во время запуска контейнера. После запуска Docker в сессии терминала запустите оболочку контейнера (bash) командой **docker run image_name: tag_name**. Имя тега можно не указывать, тогда задействуется текущая версия образа.

Введение

Вариант с **Dockerfile** сложнее.

Вы уже знаете, что каждому образу присваивается свой **Dockerfile**. После указания нужных команд в **Dockerfile**, исключите в **.dockerignore** все файлы, не используемые в сборке. Затем создайте образ командой **docker image build**, присвойте ему имя и тег.

Синтаксис команды docker run

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Опции:

--add-host – добавьте настраиваемое сопоставление хоста с IP (хост: ip);

--attach , -a – прикрепить к STDIN, STDOUT или STDERR;

--blkio-weight – блок ввода-вывода (относительный вес) от 10 до 1000 или 0 для отключения (по умолчанию 0);

--blkio-weight-device – вес блока ввода-вывода (относительный вес устройства);

--cap-add – добавить возможности Linux;

--cap-drop – удалите возможности Linux;

--cgroup-parent – необязательная родительская группа для контейнера;

Синтаксис команды docker run

--cgroupns – API 1.41+;

Используемое пространство имен Cgroup (host | private) 'host': запустите контейнер в пространстве имен cgroup хоста Docker 'private': запустите контейнер в его собственном частном пространстве имен cgroup '': используйте пространство имен cgroup, настроенное параметром default-cgroupns-mode в демоне (по умолчанию)

--cidfile – запишите идентификатор контейнера в файл;

--cpu-count – количество процессоров (только для Windows);

--cpu-percent – процент процессора (только для Windows);

--cpu-period – ограничить период CFS процессора (полностью честный планировщик);

--cpu-quota – ограничить квоту CPU CFS (полностью честный планировщик);

--cpu-rt-period – ограничить период реального времени процессора в микросекундах;

--cpu-rt-runtime – ограничить время выполнения процессора в режиме реального времени в микросекундах;

--cpu-shares , -c – доли процессора (относительный вес);

Синтаксис команды docker run

--cpus – количество процессоров;

--cpuset-cpus – процессоры, в которых разрешено выполнение (0-3, 0,1);

--cpuset-mems – MEMs, в которых разрешено выполнение (0-3, 0,1);

--detach , -d – запустите контейнер в фоновом режиме и распечатайте идентификатор контейнера;

--detach-keys – переопределить последовательность клавиш для отсоединения контейнера;

--device – добавьте хост-устройство в контейнер;

--device-cgroup-rule – добавьте правило в список разрешенных устройств cgroup;

--device-read-bps – ограничение скорости чтения (байт в секунду) с устройства;

--device-read-iops – ограничить скорость чтения (ввода-вывода в секунду) с устройства

--device-write-bps – ограничить скорость записи (байт в секунду) на устройство;

--device-write-iops – ограничить скорость записи (ввода-вывода в секунду) на устройство;

--disable-content-trust true – пропустить проверку изображения;

Синтаксис команды docker run

--dns – настройка пользовательских DNS-серверов;

--dns-opt – настройка параметров DNS;

--dns-option – настройка параметров DNS;

--dns-search – настройка пользовательских доменов поиска DNS;

--domainname – доменное имя контейнера NIS;

--entrypoint – перезаписать НАЧАЛЬНУЮ ТОЧКУ изображения по умолчанию;

--env , -e – установка переменных среды;

--env-file – чтение в файле переменных среды;

--expose – предоставить доступ к порту или диапазону портов;

--gpus – API 1.40+.

Графические устройства для добавления в контейнер ('all' для передачи всех графических процессоров)

Синтаксис команды docker run

--group-add – добавьте дополнительные группы для присоединения;

--health-cmd – команда для запуска для проверки работоспособности;

--health-interval – время между выполнением проверки (мс | с | м | ч) (по умолчанию 0 секунд);

--health-retries – последовательные сбои, необходимые для сообщения о неработоспособности;

--health-start-period – начальный период инициализации контейнера перед запуском обратного отсчета попыток работоспособности (мс | с | м | ч) (по умолчанию 0 секунд);

--health-timeout – максимальное время, необходимое для выполнения одной проверки (мс | с | м | ч) (по умолчанию 0 секунд);

--help – использование печати;

--hostname , -h – имя хоста контейнера;

--init – запустите init внутри контейнера, который пересылает сигналы и обрабатывает процессы;

--interactive , -i – держите стандартный интерфейс открытым, даже если он не подключен;

--io-maxbandwidth – максимальный предел пропускной способности ввода-вывода для системного диска (только для Windows);

--io-maxiops – максимальный предел операций ввода-вывода для системного диска (только для Windows).

Синтаксис команды docker run

--ip – адрес IPv4 (например, 172.30.100.104);

--ip6 – адрес IPv6 (например, 2001: db8::33);

--ipc – режим IPC для использования;

--isolation – технология изоляции контейнеров;

--kernel-memory – ограничение памяти ядра;

--label , -l – установка метаданных в контейнере;

--label-file – чтение в файле меток с разделителями строк;

--link – добавить ссылку на другой контейнер;

--link-local-ip – ссылка на контейнер IPv4 / IPv6-локальные адреса;

--log-driver – протоколирование драйвера для контейнера;

--log-opt – параметры драйвера журнала;

--mac-address – MAC-адрес контейнера (например, 92: d0: c6:0a:29:33);

--memory , -m – ограничение памяти;

--memory-reservation – мягкое ограничение памяти;

--memory-swap – ограничение подкачки, равное памяти плюс подкачка: '-1' для включения неограниченной подкачки;

--memory-swappiness -1 – настройка подкачки памяти контейнера (от 0 до 100);

--mount – прикрепите к контейнеру монтирование файловой системы;

--name – присвойте имя контейнеру;

Синтаксис команды docker run

--net – подключите контейнер к сети;

--net-alias – добавьте псевдоним в сетевой области для контейнера;

--network – подключите контейнер к сети;

--network-alias – добавьте псевдоним в сетевой области для контейнера;

--no-healthcheck – отключите любую проверку РАБОТОСПОСОБНОСТИ, указанную в контейнере;

--oom-kill-disable – отключить OOM Убийца;

--oom-score-adj – настройте параметры OOM хоста (от -1000 до 1000);

--pid – пространство имен PID для использования;

--pids-limit – настройте ограничение pids контейнера (установите -1 для неограниченного);

--platform – установите платформу, если сервер поддерживает мультиплатформенность;

--privileged – предоставьте расширенные привилегии этому контейнеру;

--publish , -p – опубликуйте порты контейнера на хосте;

--publish-all , -P – опубликовать все открытые порты на случайные порты;

--pull missing – извлеките изображение перед запуском ("всегда" | "отсутствует" |"никогда");

--read-only – смонтируйте корневую файловую систему контейнера как доступную только для чтения;

--restart no – перезапустите политику, которая будет применяться при выходе из контейнера;

Синтаксис команды docker run

--rm – автоматическое удаление контейнера при его завершении;

--runtime – среда выполнения, используемая для этого контейнера;

--security-opt – параметры безопасности;

--shm-size – размер /dev/shm;

--sig-proxy true – прокси-сервер получил сигналы для процесса;

--stop-signal SIGTERM – сигнал для остановки контейнера;

--stop-timeout – время ожидания (в секундах) для остановки контейнера;

--storage-opt – параметры драйвера хранилища для контейнера;

--sysctl – параметры Sysctl;

Синтаксис команды docker run

--tmpfs – смонтировать каталог tmpfs;
--tty , -t – выделите псевдо-TTY;
--ulimit – ограничивать параметры;
--user , -u – имя пользователя или UID (формат: <имя |uid>[:<группа | gid>]);
--userns – пользовательское пространство имен для использования;
--uts – пространство имен UTS для использования;
--volume , -v – привязать смонтировать том;
--volume-driver – дополнительный драйвер тома для контейнера;
--volumes-from – монтируйте тома из указанных контейнеров;
--workdir , -w – рабочий каталог внутри контейнера.