GeekBrains

Linux

Введение в Docker

На этом уроке

1. Установим Docker.
2. Познакомимся со структурой Docker и основными командами управления.
3. Познакомимся с Docker Compose.

Оглавление

Глоссарий

Контейнеризация — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Эти экземпляры с точки зрения пользователя полностью идентичны отдельному экземпляру операционной системы. Простыми словами, контейнеризация позволяет виртуализировать процесс.

Docker Hub — платформа для распространения Docker-контейнеров и управления ими.

Dockerfile — это сценарий, который состоит из последовательности команд и аргументов, необходимых для создания образа. Такие сценарии упрощают развёртывание и процесс подготовки приложения к запуску.

YAML — «дружественный» формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования.

Установка Docker

Docker лучше всего устанавливать из официального репозитория Docker. Для этого выполним ряд рекомендаций. Все действия производим от имени суперпользователя.

Сначала установим пакеты, необходимые для работы apt по протоколу HTTPS:

```
apt-get install apt-transport-https ca-certificates curl gnupg-agent
software-properties-common -y
```

Добавляем ключ репозитория:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
```

Подключаем репозиторий:

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
```

Обновляем список пакетов: `apt update`. И устанавливаем пакет: `apt install docker-ce -y`.

На данный момент Docker плотно вливается в структуру стандартных пакетов ОС Linux. Например, в последней доступной версии Debian 10 Buster свежие пакеты Docker и Docker Compose уже доступны в стандартном репозитории. При установке нет необходимости подключать дополнительный репозиторий, их можно установить, выполнив `apt install -y docker docker-compose`.

.

Обзор и структура Docker

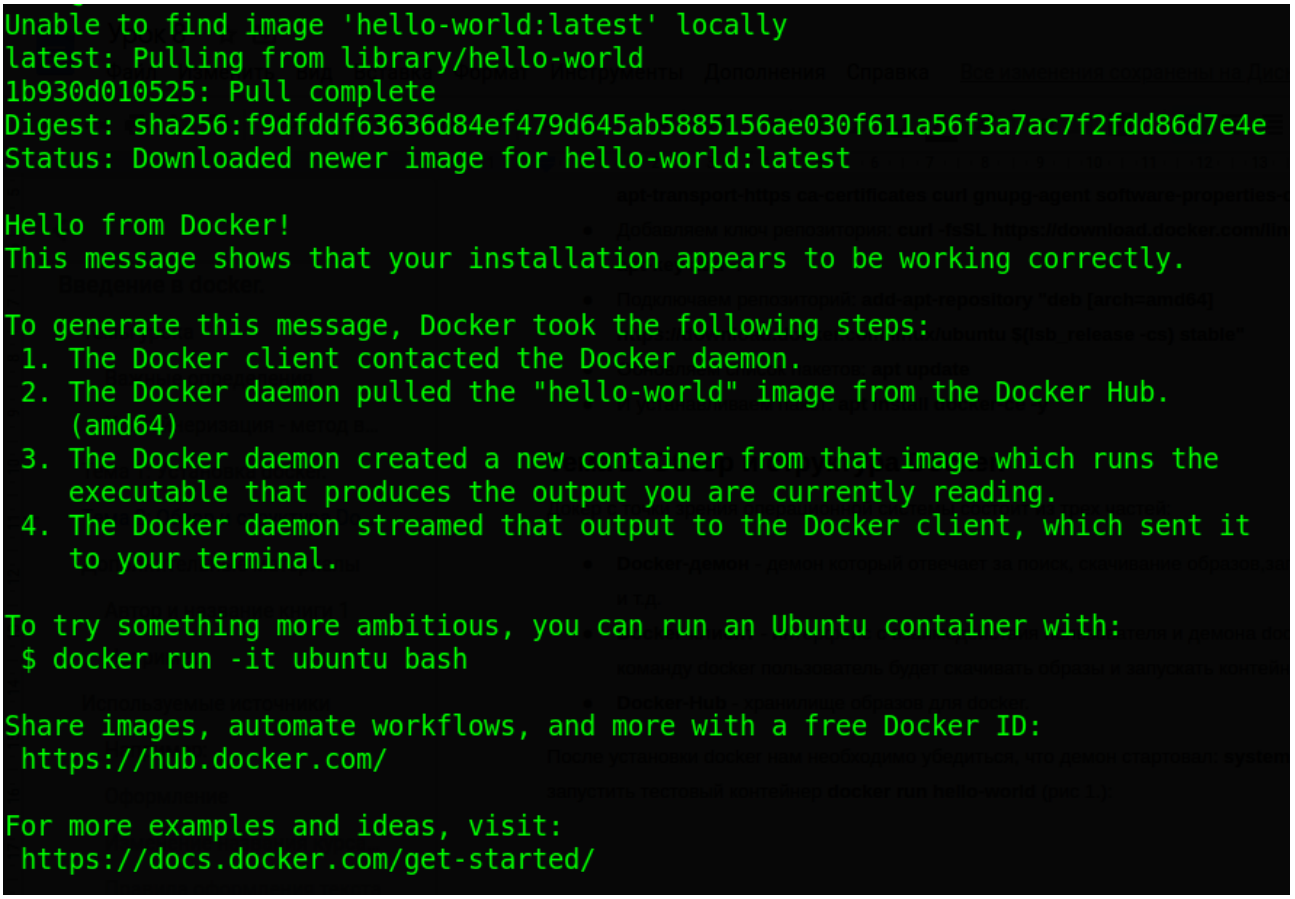
Docker с точки зрения операционной системы состоит из трёх частей:

1. **Docker-демон** — процесс, который отвечает за поиск, скачивание образов, запуск контейнеров и т. д.
2. **Docker-клиент** — интерфейс взаимодействия пользователя и демона Docker. Именно через команду Docker пользователь будет скачивать образы и запускать контейнеры.
3. **Docker Hub** — хранилище образов для Docker.

С точки зрения архитектуры Docker состоит из следующих компонентов:

1. **Images (образы)** — это своеобразный шаблон, который содержит экземпляр операционной системы с набором библиотек, необходимых для работы приложения.
2. **Registry (реестр)** — публичное или закрытое хранилище образов. Пример публичного реестра образов — Docker Hub.
3. **Container (контейнер)** — запущенное приложение, которое создано из образа.

После установки Docker нам нужно убедиться, что демон стартовал: `systemctl status docker`, и запустить тестовый контейнер **docker run hello-world**:



Загрузка образа и запуск контейнера состоят из нескольких стадий:

1. Поиск образа в локальном кеше. Если образ был уже скачан, то Docker сразу же запустит контейнер.
2. Поиск и скачивание образа из Docker Hub.
3. Запуск контейнера.

Управление образами и контейнерами

Рассмотрим базовые команды, которые нужны для управления образами и контейнерами.

Docker search image_name — поиск образа в реестре. Например, `docker search nginx` найдёт все образы, которые содержат веб-сервис nginx.

Docker pull image_name скачает диск из реестра, например, `docker pull nginx`.

Запустить контейнер из скачанного образа позволит команда run, которой мы можем передать ряд параметров. Для примера запустим контейнер nginx из уже скачанного образа:

```
docker run -d --name nginx -p 80:80 -v /var/www/html:/usr/share/nginx/html nginx
```

Здесь говорим Docker, что необходимо запустить контейнер в оперативной памяти (**параметр -d**), задаём контейнеру имя **nginx** (**параметр --name**) и пробрасываем порты: **-p 80:80**. Здесь важно, что слева находится локальный порт (порт нашей машины), а справа — порт контейнера.

Пробрасываем директорию с файлами. Принцип такой же, как и с проброcом портов: слева — локальная директория, а справа — директория внутри контейнера, и последний параметр — имя образа, из которого мы поднимаем наш контейнер **nginx**.

```
 -v  /var/www/html:/usr/share/nginx/html
```

Просмотреть список запущенных контейнеров мы можем командами `docker ps -a`.

Получить список всех контейнеров мы можем командой `docker container ls -a`.

Рассмотрим сборку собственного образа на примере веб-сервера nginx, установленного в Ubuntu. Для этого нам необходимо создать **Dockerfile** — сценарий, в котором будут описаны все шаги по сборке нашего приложения. В простейшем случае этот файл будет содержать в себе следующие директивы:

1. **FROM** определит базовый образ, из которого будет собираться контейнер. В нашем случае мы используем последнюю версию Ubuntu. Директива будет выглядеть вот так: `FROM ubuntu:latest`. При запуске Docker обратится в Docker Hub и скачает последний стабильный образ Ubuntu.
2. **MAINTAINER** сообщит контейнеру имя автора создаваемого образа.
3. **RUN** запустит команду внутри образа. В нашем случае необходима для обновления списка пакетов и установки пакета nginx:

```
RUN apt update
RUN apt install nginx -y
```

4. **ADD** берёт файлы с хоста и кладёт внутрь образа. Например, нам надо положить конфигурационный файл nginx или другие файлы в каталог приложения. В нашем примере эту директиву использовать не будем, оставим всё по умолчанию
5. **VOLUME** — директория, которая будет подключена в контейнер. В нашем случае это каталог /var/www/html, в котором будут храниться файлы нашего сайта.
6. **EXPOSE** задаст порт, через который контейнер будет общаться с внешним миром. В нашем примере — **EXPOSE 80**.
7. **CMD** — команда, которая будет запущена при старте контейнера из образа. В нашем случае: `/usr/sbin/nginx -g "daemon off;"` — запускаем nginx, отключив режим daemon. Более подробно параметры запуска nginx можно посмотреть на странице официальной документации.

Итоговый Dockerfile:

```
FROM ubuntu:latest
MAINTAINER User GB
RUN apt-get update
RUN apt-get install nginx -y
VOLUME "/var/www/html"
EXPOSE 80
CMD /usr/sbin/nginx -g "daemon off;"
```

Запускаем сборку: `docker build -t main_image_nginx .` — здесь мы говорим Docker собрать образ с именем **main_image_nginx**.

И запускаем контейнер из собранного нами образа: `docker run -d --name conteiner_name -p 80:80 image-name:latest`.

Управление сетями в Docker

Сеть в Docker реализована посредством четырёх сетевых драйверов. Можно провести аналогию с типом подключения в VirtualBox:

1. **Bridge** — сети по умолчанию, аналог типа подключения NAT в VirtualBox. Связь устанавливается через Bridge-интерфейс, который поднимается в операционной системе при установке Docker и носит название Docker0. Этот интерфейс можно увидеть, выполнив команду `ip -a`.
2. **Host** — с помощью этого драйвера контейнер получает доступ к собственному интерфейсу хоста. Аналог подключения «Мост» в VirtualBox.
3. **Macvlan** даёт контейнерам прямой доступ к интерфейсу и суб-интерфейсу (VLAN) хоста.

4. **Overlay** позволяет строить сети на нескольких хостах с Docker.

Просмотреть доступные сети можно командой `docker network ls`, а просмотреть участников сети можно, выполнив `docker network inspect network_name`. Например, `docker network inspect bridge` покажет контейнеры, которые работают в этой сети.

Доступ к приложениям, запущенным в контейнере, осуществляется через **iptables**. После установки Docker создаёт ряд правил в таблице filter, посредством которых осуществляется фильтрация трафика, и правила в таблице NAT, через которые происходит проброс портов до поднятых приложений.

Docker-Compose

docker-compose повторяет весь функционал **Docker**, за исключением одного: если Docker применяется для управления одним конкретным сервисом, то docker-compose позволяет управлять несколькими контейнерами, входящими в состав приложения.

Установка Docker Compose проста. Нет необходимости подключать какие-то дополнительные репозитории, достаточно выполнить две команды согласно рекомендациям официального сайта. Скачать текущий стабильный релиз:

```
curl -L
"https://github.com/docker/compose/releases/download/1.25.4/docker-compose-$(una
me -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

И выставить права на исполнение: `chmod +x /usr/local/bin/docker-compose`.

Соберём контейнер с nginx, используя Docker Compose. Для сборки и запуска контейнера Docker Compose использует синтаксис YAML. Создадим простой проект наподобие примера выше, только будем использовать уже готовый образ nginx из Docker Hub. Для этого, используя консольный редактор, создаём файл docker-compose.yml и вносим в него следующее содержимое:

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - 80:80
    volumes:
      - /var/www/html
```

Здесь строка `version '3'` говорит об использовании третьей версии формата файлов для Docker Compose. Директива service описывает службу, которую мы будем запускать, дальше идёт имя nginx. Собираем контейнер из последней стабильной версии nginx, доступной на Docker Hub: `image: nginx:latest` и пробрасываем 80-й порт хост-машины и каталог /var/www/html, используя директивы ports и volumes.

Аналогичным образом мы вызывали команду docker с соответствующими параметрами. Обратите внимание, что файл docker-compose.yml для каждого контейнера должен лежать в отдельной папке.

Запускаем наш проект, используя команду `docker-compose up -d --build`, — говорим, что Docker Compose должен запустить контейнер в оперативной памяти, выполнив сборку из образа.

Практическое задание

1. Запустить контейнер с Ubuntu.
2. * Используя Dockerfile, собрать связку nginx + PHP-FPM в одном контейнере.

Дополнительные материалы

Установка Docker в Ubuntu

Установка Docker Compose

Compose file reference

Используемые источники

Введение в Docker

Шпаргалка по командам Docker

Docker и сети

Docker Compose