Skillbox

Docker Compose

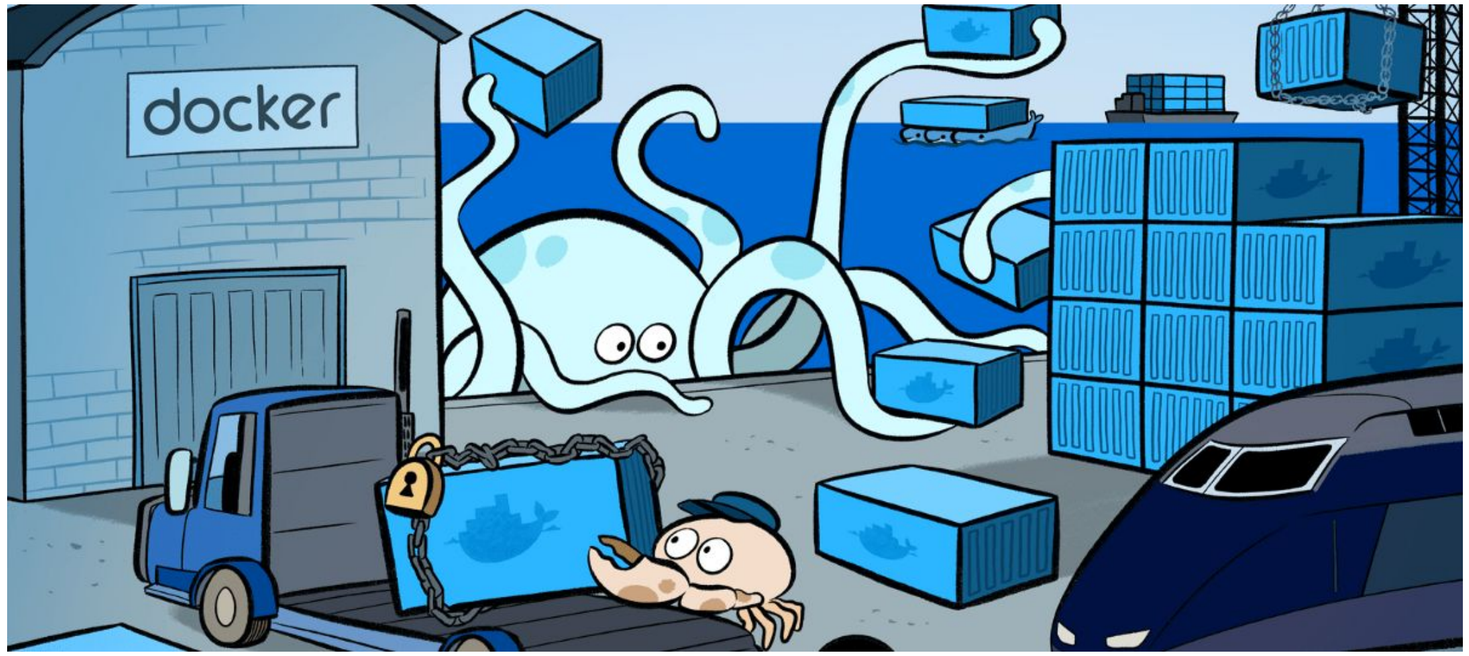
Что такое Docker Compose?

- **Docker Compose** — это инструментальное средство, входящее в состав Docker
- Оно предназначено для решения задач, связанных с развёртыванием проектов

Разница между Docker, Dockerfile и Docker Compose

- Docker применяется для управления отдельными контейнерами (сервисами), из которых состоит приложение
- Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения. Этот инструмент предлагает те же возможности, что и Docker, но позволяет работать с более сложными приложениями
- Docker (отдельный контейнер) и Docker Compose (несколько контейнеров)
- Dockerfile описывает, как построить образ Docker - это, по сути, версия Docker Makefile или pom.xml или build.gradle
- С помощью Dockerfile мы определяем среду приложения, docker-compose.yml определяет сервисы приложения, docker-compose up, чтобы запустить приложение

Пример сценария использования docker-compose



Создание проекта

Нам потребуется:

- файл docker-compose.yml. Это файл Docker Compose, который будет содержать инструкции, необходимые для запуска и настройки сервисов
- папка server. Она будет содержать файлы, необходимые для обеспечения работы сервера
- папка client. Здесь будут находиться файлы клиентского приложения

Результат:

```
├── client/
├── docker-compose.yml
└── server/

2 directories, 1 file
```

Создание сервера

Перейдите в папку server и создайте в ней следующие файлы:

- файл server.py. В нём будет находиться код сервера
- файл index.html. В этом файле будет находиться фрагмент текста, который должно вывести клиентское приложение
- файл Dockerfile. Это файл Docker, который будет содержать инструкции, необходимые для создания окружения сервера

Вот как должно выглядеть содержимое вашей папки server/:.

```
├── Dockerfile
├── index.html
└── server.py

0 directories, 3 files
```

Создание сервера

Добавим в файл server.py следующий код:

```
#!/usr/bin/env python3
import http.server
import socketserver
handler = http.server.SimpleHTTPRequestHandler
with socketserver.TCPServer(("", 1234), handler) as httpd:
httpd.serve_forever()
```

Создание сервера

В файл index.html добавим следующий текст:

Docker-Compose is magic!

Dockerfile:

FROM python:latest

ADD server.py /server/

ADD index.html /server/

WORKDIR /server/

Создание клиента

Перейдите в папку вашего проекта client и создайте в ней следующие файлы:

- файл client.py. Тут будет находиться код клиента
- файл Dockerfile. Этот файл играет ту же роль, что и аналогичный файл в папке сервера. А именно, он содержит инструкцию, описывающую создание среды для выполнения клиентского кода

В результате ваша папка client/ на данном этапе работы должна выглядеть так:

```
├── client.py
└── Dockerfile

0 directories, 2 files
```

Создание клиента

Добавим в файл client.py следующий код:

```
#!/usr/bin/env python3
import urllib.request
fp = urllib.request.urlopen("http://localhost:1234/")
encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")
print(decodedContent)
fp.close()
```

Создание клиента

Как и в случае с сервером, мы создаём для клиента простой Dockerfile, ответственный за формирование среды, в которой будет работать клиентское Python-приложение:

Dockerfile:

```
FROM python:latest
ADD client.py /client/
WORKDIR /client/
```

Docker-compose

1

docker-compose.yml:

```
version: "3"
services:
  server:
    build: server/
    command: python ./server.py
    ports:
      - 1234:1234
```

2

```
client:
    build: client/
    command: python ./client.py
    network_mode: host
    depends_on:
      - server
```

Сборка и запуск проекта

Сборка проекта:

```
$ docker-compose build
```

Запуск проекта:

```
$ docker-compose up
```

http://localhost:1234/ = Docker-Compose is magic!

Полезные команды

Останавливать и удалять контейнеры, созданные docker-compose up:

```
$ docker-compose down
```

Вывод журналов сервисов:

```
$ docker-compose logs -f [service name]
```

Список контейнеров:

```
$ docker-compose ps
```

Выполнить команду в выполняющемся контейнере:

```
$ docker-compose exec [service name] [command]
```

Например, docker-compose exec server ls.

Вывести список образов:

```
$ docker-compose images
```

Итоги

- Узнали, что такое Docker Compose
- Разобрали разницу Docker, Dockerfile, Docker Compose
- Рассмотрели пример с использованием docker-compose

Спасибо
за внимание!