概述

- 传统方式：开发人员（Developer） ---- 打包好的应用 ----> 运维人员（Operator）
  - 专职的运维人员负责
    - 配置环境：硬件 + 软件
    - 部署应用
    - **问题：环境不一致导致失败**
      - 版本问题，操作系统...
- 解决办法：开发人员 ---- 镜像（环境 + 打包好的应用） -----> 运维人员
  - 运维人员只需运行容器就可以
  - 依然存在其他问题（教材第12章）
    - 对于运维人员：采购，管理，维护硬件
      - 解决办法：完全交给云
    - 对于开发人员：创建管理容器
      - 解决办法：云原生（Serverless）
- **Docker 是一个跨平台的、可移植的、简单可用的容器解决方案**

Docker组件

- 容器（Container）
  - 包含了应用及其需要的环境
  - 是应用的运行时表现形式
- 镜像（Image）
  - 创建容器的基础
  - 镜像是容器的模板：类似于 OOP 中的类
  - 容器是镜像的实例：类似于 OOP 中的对象
- 标签（Tag）
  - 镜像版本号
  - mysql:latest = mysql，mysql:8
- 仓库（Repository）
  - 存放镜像的地方
    - Linux，Windows，JDK， Python，WordPress，MySQL ...
  - 推送（Push）与拉取（Pull）
  - 官方仓库：https://hub.docker.com/

- Dockerfile
  - 镜像描述文件
  - 基于 Dockerfile 生成镜像
- docker-compose.yml
  - 容器编排，同时管理多个容器
  - 同时创建，运行，销毁多个容器

实战：安装 Docker Desktop for Windows

- 环境说明
  - Windows 10（升级到最新版本）
  - CPU支持虚拟化：CPUZ
- 内存至少4GB
- 过程简介
  - 下载 Docker Desktop：
    - https://hub.docker.com/editions/community/docker-ce-desktop-windows
    - 已放在 QQ 学习群中
  - 安装：
    - 一直点击下一步
    - 中间会提示安装 Windows 虚拟化功能，
    - 安装完后会重启
  - 配置加速器：加快镜像摘取的速度
    - 网易云：https://hub-mirror.c.163.com
    - 华中科技大学：https://docker.mirrors.ustc.edu.cn
  - 运行：出现在右下角通知区
  - 测试：打开 PowerShell ，运行命令 `docker run hello-world`

实战：部署 MySQL 服务器

- 把别人做好的镜像直接拿过来用，对应的场景有：
  - 我需要一个服务器
  - 我想学习 Linux 操作系统
  - 我想搭建自己的个人博客
- 操作过程
  - 打开 MySQL 镜像官方发布页面
    - https://hub.docker.com/_/mysql/
  - 选择合适的版本（标签）
    - latest，8，5
  - 拉取镜像：选择其中一个
    - docker pull mysql
    - docker pull mysql:8
    - docker pull mysql:5.7
  - 检查镜像是否拉取成功
    - docker images
  - 创建并运行容器

        - docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -d mysql
        - --name：给容器起个名字，可以不写，Docker会随机生成一个名字
        - -e MYSQL_ROOT_PASSWORD=root：设置 root 密码的环境变量，密码为 root
        - -d 后台运行
        - -p 3306:3306：把容器中的MySQL的3306端口映射到主机的3306端口
            - 不映射，无法访问
    - 查看容器是否运行
        - docker ps [-a]
    - 打开 MySQL 客户端，进行连接，并查看效果

实战：部署后端应用到容器

- 找不到现成的，制作自己的镜像
- 使用 Spring Boot 开发一个简单的API

```
@CorsCrossing("*")
@RestController
@SpringBootApplication
public class Application {
    @GetMapping("/hello/{name}")
    public String hello(@PathVariable String name) {
        return String.format("Hello %s!", name);
    }
}
```

- 打包（jar包）并运行，保证在本地可以跑
    - 打包：mvn package
    - 运行：java -jar hello.jar
- 编写 Dockerfile
    - 官方Dockerfile指南：https://docs.docker.com/engine/reference/builder/

```
FROM openjdk:11                             # 在 jdk8 镜像的基础上进行构建
ADD hello.jar /app.jar                      # 把 hello.jar 放到容器的根目录下并改名为 app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]  # 在容器中运行应用
```

- 构建自己的应用：
    - 把 Dockerfile 和 hello.jar 放在同一个目录中，假设为 docker

```
PS> cd docker
PS> docker build -t spring-hello . # 构建镜像 spring-hello，等价于spring-hello:latest
PS> docker images                  # 查看镜像是否生成成功
PS> docker run -p 80:8080 -d spring-hello
```

- 打开浏览器，输入地址：`http://localhost:80/hello/docker`，检查结果是否正确

实战：操作多个容器

- 问题
  - 多个容器共同协作共同组成一些应用
    - 每个容器都有自己的配置
    - 容器间可能相互隔离
    - 容器间也可能存在依赖关系
    - 容器间也有可能共享配置
  - 对于大型应用，**以容器为单位进行部署不可取的**
    - 低效、容易出错
- 解决办法：服务（容器）编排
  - **以应用为单位进行部署**
  - **一键部署或停止其所有的容器**
  - docker-compose描述文件
- 开发环境介绍
  - 前端：Nodejs + Vuejs + Vs Code + Chrome
    - 验证：node -v， npm -v
    - npm下载加速：npm install -g cnpm --registry=https://registry.npm.taobao.org
    - vuejs：cnpm install -g @vue/cli
  - 后端：JDK8+、STS4.5
- 制作前端镜像
  - 使用 Vuejs 编写前端应用

```
<template>
  <div id="app">
    <div id="sender">
      <input type="text" v-model="name" />
      <button @click="send">发送</button>
    </div>
    <div id="greeting">
      <p style="margin-bottom: 0;">Vue: {{ name }}</p>
      <p style="margin-top: 0;">Spring: {{ greeting }}</p>
    </div>
  </div>
</template>

<script>
export default {
  name: "App",
  data() {
    return {
      name: "",
      greeting: ""
    };
  },
  methods: {
    send() { // Axios Promise
      fetch(`http://localhost:8000/hello/${this.name}`)
        .then(response => {
          return response.text();
        })
        .then(text => (this.greeting = text))
        .catch(error => console.log(error));
    }
```

```
  }
};
</script>
```

- 打包前端应用
  - `npm run build`
  - 生成 dist 目录
- 传统的方式部署前端应用
  - 下载 nginx 服务器
  - 把打好的包拷贝到服务器根目录下的 html 目录
  - 启动服务器并访问：http://localhost
- 编写 Dockerfile

```
FROM nginx   # nginx 是一个 Web 服务器，可以托管静态网站
             # 更多请参考https://hub.docker.com/_/nginx/
COPY ./hello /usr/share/nginx/html  # 将 hello 目录下的文件拷贝到镜像中
```

- 制作前端镜像

```
PS> docker build -t vue-hello .
```

- 制作后端镜像 spring-hello并测试
  - 与上一节开发的后端应用相比：
    - 跨域访问的处理 @CorsCrossing("*")
    - 修改端口为8000

```
PS> docker build -t spring-hello .
PS> docker run --name hello-backend -d -p 8000:8080 spring-hello
PS> Invoke-RestMethod http://localhost:8000/hello/lisi # 使用 PowerShell 的命令调用接口
```

- 启动前端容器并测试

```
PS> docker run --name hello-frontend -d -p 80:8080 vue-hello
PS> curl http://localhost/hello/lisi
# 或 在浏览器中访问http://localhost:80/hello/lisi
```

- 编写 docker-compose.yml 文件
  - 提示：此文件最好放在一个英文名称的目录下
  - docker-compose官方指南：https://docs.docker.com/compose/compose-file/

```yaml
version: '3'
services:
  frontend:
    image: vue-hello
    ports:
    - 80:80
    depends_on:
    - backend
  backend:
    image: spring-hello
    ports:
    - 8000:8000
```

  - 部署应用：`docker-compose up`
  - 卸载应用：`docker-compose down`
  - DevOps

参考资料

- 周立，Spring Cloud 与 Docker 微服务实战，第2版，电子 工业出版社, 2018.7
  - 主要参考 Docker 相关的三章

作业与实验

- （作业）结合自己的生活学习实际，谈一谈 Docker 这样一项技术能解决你或身边同学的什么问题。最好能举例说明。回答要分为三部分：
  - 问题描述：描述你在生活学习中真正存在的痛点，请认真思考。
  - 解决思路：从原理思想上概要性描述为什么Docker技术能够解决。
  - 解决过程：根据你的思路把解决过程描述出来。此部分不是实际操作，不用截图，不用写代码，只需要文字描述把每步要做的事概括出来就来

  大家尽量认真写，这是锻炼大家 "发现问题，并使用所学技术解决问题" 的好机会。字数我不做硬性要求，但应该短不了。另外不要上传附件，把文字贴到学习通上就行。
- （选做）最新的 Docker Desktop 已经支持与 WSL 2的集成。在自己的计算机上安装 WSL 2，并把 Docker 引擎从 Hyperv 虚拟机切换到 WSL 2。完成后，把整个过程记录下来，并分析基于 WSL 2 的安装方式比虚拟机的安装方式有什么优势。
- （实验1）phpmyadmin是一个著名的基于网页的MySQL客户端，请使用 Docker 技术，在自己的电脑上部署一台 phpmyadmin 的服务器，另外重复课堂中的操作，使用 Docker 技术在自己的电脑上部署一台 MySQL 服务器，然后使用（宿主机上的浏览器打开） phpmyadmin 连接并操作 MySQL 服务器。记录最后效果的截图，以及遇到的问题。
- （实验2）使用 Spring Boot 开发一个小型的应用程序，一个接口即可，题目可自拟。开发结束后制作镜像并部署到容器中进行测试。记录实验过程和结果，以及遇到的问题及解决方法。
- （实验3）为实验2配置一个前端应用并制作镜像，然后使用 docker-compose.yml 描述并发布你的整个应用。