Docker使用教程

Docker的定义

Docker是一个开源的容器化平台，它允许开发者将应用及其依赖打包到一个可移植的容器中，从而在任何支持Docker的环境中运行这些应用。确保了在开发环境、测试环境、生成环境中能够保持一致运行的状态。

docker采用容器化技术，它比传统虚拟机更加轻量，直接运行在宿主的主机内核之上，不需要额外的操作系统，从而大大提升了性能和资源利用率。

Docker的核心概念

- **镜像（Image）**
  - 镜像是用于创建容器的模板，它包含了运行应用程序时所需的所有内容，如代码、库、环境变量等。
  - 它通过**dockerfile文件**构建，创建后只读不可修改，类似于操作系统的ISO文件。
  - 可以把它打包成tar文件，也可以把它上传到Docker Hub，供其他人下载运行。

- **容器（Container）**
  - 容器是镜像运行的实例，是一个能够独立运行的应用程序。容器提供了一种轻量级的虚拟化方式，使得应用程序能够在隔离的环境中运行。
  - 运行时共享宿主机的资源，如CPU、内存、网络等。
  - 容器启动速度快、占用资源少、避免了传统虚拟机的臃肿。它可以运行、停止、重启、销毁。

- **仓库（Repository）**
  - 仓库是用于储存和分发Docker镜像的地方。公共仓库例如Docker Hub，提供了大量官方和社区镜像。用户也可以搭建自己的私有仓库，储存自己的镜像。
  - 用户可以通过推送（push）和拉去（pull）来管理镜像。

- **Docker Compose**
  - compose是一个用来定义和运行多个容器的工具。通过compose，使用YAML文件来配置应用程序所需要的所有服务（`docker-compose.yml`）。
  - `docker-compose.yml` **文件示例**

```
# 指定 Docker Compose 文件的版本。在这个例子中，使用的是版本 3。这
# 不同版本可能支持不同的特性。
version: '3'

# 用于定义应用程序中包含的服务（容器）。每个服务都可以有自己的配置，
services:

  # 这是第一个服务的名称，名为 web。服务名称可以任意命名，但通常建
 web:

   # 指示 Docker Compose 从当前目录（.）构建镜像。这意味着在当前目
  build: .

  # 用于定义容器端口与宿主机端口之间的映射。在这个例子中：
    # "5000:5000" 表示将宿主机的 5000 端口映射到容器的 5000 端口。
  ports:
   - "5000:5000"

 # 这是第二个服务的名称，名为 redis。这个服务使用 Redis 数据库。
 redis:

   # 指定要使用的 Docker 镜像。在这个例子中，使用的是 redis:alpine 镜
  image: "redis:alpine"

  # 总结
    # 这个 docker-compose.yml 文件定义了两个服务：一个是基于当前目
```

- **Dockerfile（固定文件名）**

- Dockerfile是一个用于构建Docker镜像的文本文件，它包含了一系列指令，如：指定基础镜像、运行命令、拷贝文件、安装依赖等内容。（如: FROM、COPY、RUN）
- 在项目根目录下创建该文件：Dockerfile

```
# 1. 指定基础镜像
FROM python:3.9

# 2. 维护者信息（可选）
LABEL maintainer="your_email@example.com"

# 3. 设置工作目录
WORKDIR /app

# 4. 复制项目文件到容器
COPY . .

# 5. 安装依赖
RUN pip install -r requirements.txt

# 6. 暴露端口
EXPOSE 5000

# 7. 运行应用
CMD ["python", "app.py"]
```

```
FROM  指定基础镜像（必须是第一行指令）
LABEL 添加元数据，如作者信息等
WORKDIR  指定容器内的工作目录
COPY  复制本地文件到容器
ADD   和 COPY 类似，但支持自动解压 .tar.gz
RUN   在构建阶段运行命令（如安装软件）
EXPOSE  声明容器的对外端口（仅作文档用途）
ENV 设置环境变量
CMD   指定容器启动时运行的默认命令（可被 docker run 覆盖）
```

```
ENTRYPOINT 指定容器启动时运行的主命令（不可被 docker run 覆盖）
VOLUME 定义数据卷（持久化存储）
```

Docker的常用命令

- **镜像（Image）常用命令**

```
# 构建docker镜像
# -t myflaskapp 指定镜像名称
# . 表示当前目录（Dockerfile 所在目录）
docker build -t <自定义镜像名称> .

# 查看本地已有的镜像
docker images

# 拉取远程镜像
docker pull 镜像名

# 通过 Dockerfile 构建镜像
docker build -t 镜像名 .

# 删除本地镜像
docker rmi 镜像ID

# 将镜像导出，给他人使用。-o 是output的缩写
# 示例：docker save -o my-docker.tar hello-docker
docker save -o <自定义tar文件名> <镜像名>

# 加载镜像，-i是--input的缩写。加载完毕后，使用docker images查看是否加载成
docker load -i my-docker.tar
```

- **容器（Container）常用命令**

```
# 创建容器
# -d 后台运行模式(如果不加 -d，容器会运行在前台，终端会一直被占用，直到手动
# -p 端口映射，Publish Ports(用于将宿主机端口映射到容器端口，使容器的服务可

docker run -d -p 5000:5000 --name <自定义容器名字> <镜像名字>

docker ps                    # 查看正在运行的容器
docker ps -a                 # 查看所有容器，包括没有运行的
docker stop <容器ID或容器名称>      # 停止某个容器
docker start <容器ID或容器名称>     # 启动某个容器
docker restart <容器ID或容器名称>   # 重启某个容器
```

- **Compose常用命令**

```
docker compose up -d        # 后台运行容器
docker compose ps           # 查看当前运行的容器
docker compose logs         # 查看容器日志
docker compose start        # 启动已停止的容器
docker compose stop         # 停止运行的容器
docker compose restart      # 重启容器
docker compose pause        # 暂停容器
docker compose unpause      # 取消暂停容器
docker compose down         # 停止并删除所有容器

# 将compose导出，将多个镜像导出到一个.tar文件中
# 示例：docker save -o my-compose-project.tar my-app my-database my-red
docker save -o <自定义名字.tar> <镜像1> <镜像2> <镜像3>

# 导入镜像
docker load -i xxx.tar
```