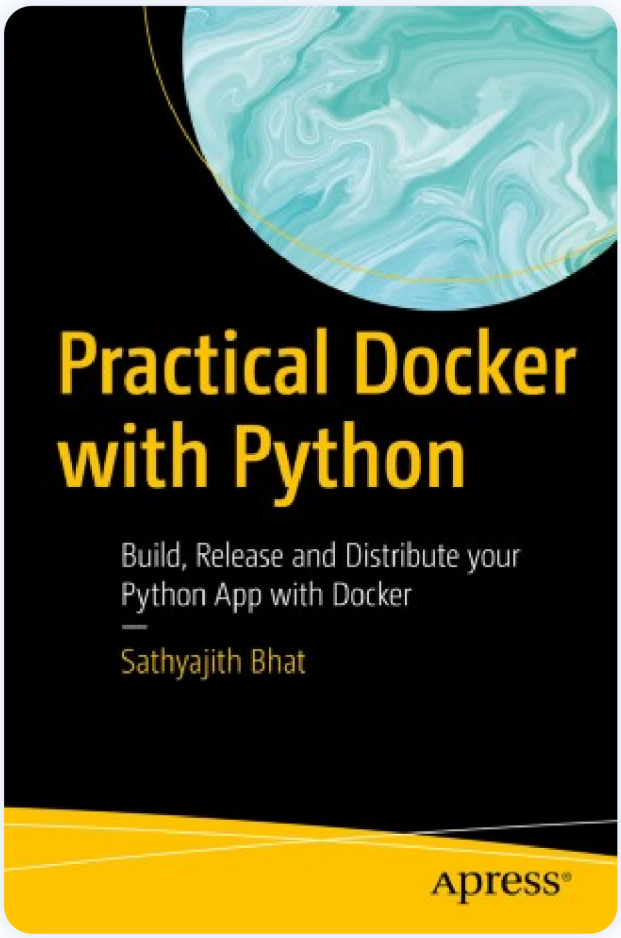


Docker

一、理解容器和虚拟机之间的差别

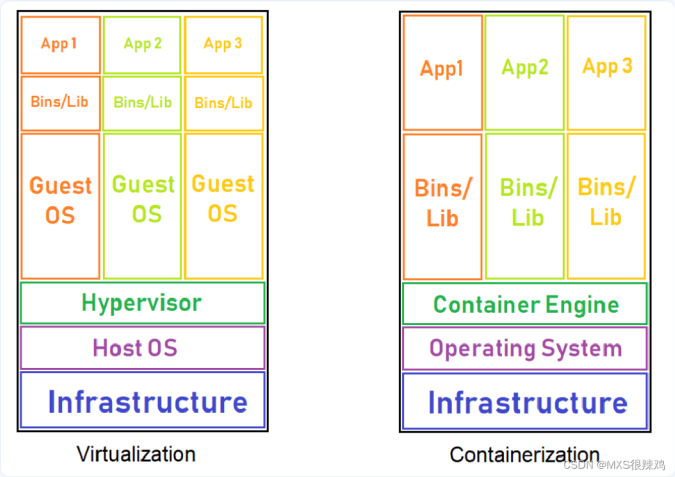
> Docker仅会隔离单一进程(或者一组进程)

- **什么是虚拟机技术**

  传统的虚拟机技术（VMware）将一组硬件虚拟化，在其上安装并运行一个完整的操作系统，然后在该系统上运行所需的应用进程；虚拟机就像一台主机。

- **什么是容器化技术**

  容器的目的是将一个应用程序及其依赖资源封装在其环境中。这种封装允许它们在隔离状态下运行，同时使用与服务器内其他容器相同的系统资源和操作系统。由于没有在运行独立的操作系统任务上浪费资源，容器化允许更快速、更有效地部署应用程序。



二、Docker入门

> Docker的安装，跟随官网指引即可（https://docs.docker.com/get-docker/）

- **理解Docker的相关术语**

  1. 层：层就是指应用于Docker镜像的一项修改，它是用Dockerfile中的一个指令来表示的
  2. 镜像：一种只读版本，是应用程序的基础，Docker镜像是是用Dockerfile中被称为指令的一系列命令来创建的
  3. 容器：一个 Docker 镜像在宿主机计算机中运行时，会产生一个具有专用命名空间的进程，这就是所谓的 Docker 容器

     进程被杀掉时并不会保存我们所进行的变更，但我们可以通过容器卷的方式绕过这一限制
  4. 绑定挂载和卷：tmpfs卷仅存储在宿主机系统的内存中，而绑定挂载和卷被存储在宿主机文件系统中
  5. Docker注册服务器：Docker Hub是一个存储Docker镜像的地方，我们可以拿来直接用
  6. Dockerfile：Dockerfile是一组指令，它会告知Docker如何构建一个镜像。一个典型的Dockerfile由以下部分组成

     FROM ENV RUN CMD

7. Docker引擎：Docker守护程序　Docker CLI　Docker API
8. Docker Compose:用于定义和运行多容器应用程序的工具
9. Docker Machine:用于在多个虚拟主机上安装Docker引擎并管理这些主机

- 常用Docker命令

```
docker pull
docker run
docker image inspect
docker ps -a
docker stop
docker kill
docker image ls
docker rm
docker rmi
```

三、理解Dockerfile

> 一个简单的Dockerfile样例
>
> ```
> FROM ubuntu
> LABEL autor="MoAn"
> LABEL description="An example Dockerfile"
> RUN apt-get install python
> COPY hello-world.py
> CMD python hello-world.py
> ```

1. 使用docker build进行构建

```
cd ./practical-docker-with-python/source-code/chapter-4/exercise-1
docker build .
```

build命令执行完毕后会反馈给我们一个容器ID

```
docker run container-ID
#我们可以给上面生成的镜像增加一个方便于记忆的名称
docker tag image_id tag_name
#可以使用docker image 查看该镜像是否标记成功
docker image
```

2. Dockerfile指令

1. FROM

```
FROM <image> [AS <name>]#每一个镜像都是从一个基础镜像开始的
```

2. WORKDIR

```
WORKDIR path#用于设置RUN,CMD,ENTRYPOINT,COPY,ADD指令的当前工作目录
```

3. ADD和COPY

```
COPY path target_path #支持基础的将文件复制到容器的功能
ADD path target_path #支持类似tarball文件自动提取和远程URL这样的特性
```

支持通配符的使用

4. RUN

```
RUN <command>
#学会打包指令
RUN apt-get update && apt-get install -y foo bar baz
```

5. CMD和ENTRYPOINT

```
CMD ["/bin/echo","This is test cmd"]
```

6. ENV

```
ENV <key>=<value>
```

7. VOLUME

```
VOLUME /var/logs/nginx
```

8. EXPOSE

```
EXPOSE <port>
```

四、理解Docker卷

> 对于docker中的容器，当我们将其删除后，重新创建一个容器会发现我们之前在容器内部做的所有更改并不会保存在镜像内，导致了所有的操作丢失，我们该如何解决这一问题

1. tmpfs挂载

   tmpfs会在tmpfs中创建一个挂载，它是一个临时文件存储设施，并不会对文件进行持久化的存储

```
docker run -it --mount type=tmpfs,target=tmpfs-mount ubuntu bash
docker run -it --tmpfs tmpfs-mount unbuntu bash
```

2. 绑定挂载

```
docker run -it -v 主机地址/容器内地址 bash
```

3. 卷

> 卷是目前最为推荐的将容器中存储的数据持久化的方法，卷是完全由Docker托管的

```
docker volume create --name=nginx-volume
docker volume inspect nginx-volume
docker volume ls
docker volume rm <name>#docker不会移除使用中的卷

#在启动容器时使用卷
docker run -it -v:nginx-volume:/data-volume
```

五、理解Docker网络

> 容器网络主要用于实现（或者限制）跨容器通信

1.默认的Docker网络通信

- bridge

  它允许所有连接到同一个网络的容器进行通信，而不处于相同桥接上的容器之间不能通信

- host

  容器就会被附加到宿主机上，如果Docker宿主机上只有这一个容器，那么是十分合适的

- overlay

  overlay网络会创建一个跨多个Docker宿主机的网络。主要用于以Swarm模式设置的Docker宿主机集群

- 其余感觉不大用的到

2.使用Docker网络

```
docker network ls
docker network inspect
docker network create <name>
#使用docker网络
docker run -d --network <name>
docker network connect <network-name> <container-name>
```

六、理解Docker Compose

Docker Compose概述

> Compose文件是一个YAML格式文件，该文件定义了启动应用程序所需要的服务，网络和卷

Docker Compose文件参考

1. Service

- build

> build键值包含了在构建时要应用的配置选项。构建上下文的路径

```
services:
    app:
        build: ./app
```

- context

> context键值用于设置构建的上下文。如果上下文是一个相对路径，那么该路径就是相对于Compose文件位置而言的

```
build:
    context: ./app
    Dockerfile: dockerfile-app
```

2. image

```
services:
    app:
        build: ./app
        image: <image/name>
```

3. environment

```
services:
    app:
    image: mysql
    environment:
        PATH: /home
        API_KEY: keyword
```

4. depends_on

> depends_on的值应该为其余服务的名称，表示只有在在依赖服务启动成功后，才可以运行该应用

5. ports

> 用于指定公开的端口号
>
> ```
> ports:
>     - "8080:80"
> ```

6. volumes

> 提供服务的命名卷
>
> ```
> volumes:
>     - "dbdata:/var/lib/mysql"
> ```

7. restart

> 可以选择的有
>
> - no:容器绝不重启
> - always:容器总在退出后重启
> - on-failure:错误导致的退出之后重启
> - unless-stopped:除非显式退出或者守护程序停止运行，否则容器总是重启

Docker Compose CLI参考

1. docker-compose up -d

> 该指令用于构建compose文件中的所有app

2. docker-compose down

> 该命令会停止容器的运行，继而移除容器、卷和网络

3. docker-compose exec

> 等同于docker exec 指令

4. docker-compose logs

> logs仅显示所有服务的最新日志

5. docker-compose stop

> stop命令会停止容器的运行

一个docker-compse实战

> 这是一个以前做过的推荐系统的项目，里面涉及到了一套完整的离线推荐和实时推荐系统，后续会将该代码构建的完整Docker文件上传至github(https://github.com/wr0519/partybuild-recommender),
>
> 但是并不建议大家这样做，对于复杂的大数据项目Docker可能会对性能产生一定的影响，同时请不要将MySQL装到Docker中。

```
version: '3'

services:
  #tomcat服务器
  tomcat:
    image: moan0/party-api:tomcat-2.0
    ports:
      - 8085:8080
    volumes:
      - ./tomcat/log:/usr/local/tomcat/user_log
      - ./tomcat:/tomcat
    networks:
      - recommend_network
```

```yaml
  #mysql服务器
  mysql:
    image: mysql:latest
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: root
    volumes:
      - ./mysql:/mysql
    networks:
      - recommend_network
  #mongo服务器
  mongo:
    image: moan0/movie-recommender:mongo1.0
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: 123456
    ports:
      - 27017:27017
    volumes:
      - ./mongodb:/mongodb
    networks:
      - recommend_network
  #mongo图形化界面
  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    volumes:
      - ./mongodb-express:/mongodb-express
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: 123456
      ME_CONFIG_MONGODB_URL: mongodb://root:123456@mongo:27017/
    networks:
      - recommend_network
  #Redis服务器
  redis:
    image: moan0/movie-recommender:redis1.0
    ports:
      - 6379:6379
    environment:
      # ALLOW_EMPTY_PASSWORD is recommended only for development.
      - ALLOW_EMPTY_PASSWORD=yes
```

```
      - REDIS_DISABLE_COMMANDS=FLUSHDB,FLUSHALL
    volumes:
      - ./redis/conf/redis.conf:/etc/redis/redis.conf
      - ./redis:/redis
    networks:
      - recommend_network
  #elasticsearch服务器
  elasticsearch:
    image: moan0/movie-recommender:elasticsearch1.0
    environment:
      - "ES_JAVA_OPTS=-Xms1024m -Xmx1024m"
      - "TZ=Asia/Shanghai"
    ports:
      - '9200:9200'
      - '9300:9300'
    restart: always
    volumes:
      - ./elasticsearch:/elasticsearch
    networks:
      - recommend_network
  #Spark环境配置
  master:
    image: bitnami/spark:3.0.2
    container_name: master
    user: root
    environment:
      - SPARK_MODE=master
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    ports:
      - '9090:8080'
      - '7077:7077'
      - '4040:4040'
    volumes:
      - ./spark-master:/spark-master
    networks:
      - recommend_network
  worker1:
    image: bitnami/spark:3.0.2
    container_name: worker1
    user: root
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://master:7077
```

```
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - ./spark-worker1:/spark-worker1
    networks:
      - recommend_network
  worker2:
    image: bitnami/spark:3.0.2
    container_name: worker2
    user: root
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://master:7077
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - ./spark-worker2:/spark-worker2
    networks:
      - recommend_network
  #flume环境配置
  flume:
    image: probablyfine/flume:2.0.0
    environment:
      - FLUME_AGENT_NAME=docker
    ports:
      - 7878:7878
    volumes:
      - ./flume/flume.conf:/opt/flume-config/flume.conf
      - ./tomcat/log/:/flume
    networks:
      - recommend_network
  #KafKa环境配置
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
     - "19005:2181"
    networks:
      - recommend_network
```

```
  kafka:
    image: wurstmeister/kafka
    hostname: kafka
    ports:
     - "19004:19004"
    environment:
     - TZ=CST-8
     - KAFKA_BROKER_ID=1
     - KAFKA_ADVERTISED_HOST_NAME=kafka
     - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
     - KAFKA_LISTENERS=INSIDE://:9092,OUTSIDE://:19004
     - KAFKA_ADVERTISED_LISTENERS=INSIDE://kafka:9092,OUTSIDE://kafka:19004
     - KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
     - KAFKA_INTER_BROKER_LISTENER_NAME=INSIDE
    volumes:
     - ./kafka:/kafka
    links:
     - zookeeper
    networks:
      - recommend_network
  #Azkaban配置
  azkaban-web:
    image: moan0/party-api:azkaban-web-1
    user: "hadoop:hadoop"
    container_name: party-azkaban-web
    hostname: azkaban-web
    restart: always
    privileged: true
    depends_on:
      - azkaban-exec
    env_file:
      - .env
    volumes:
      - ./conf/web/azkaban.properties:${AZKABAN_HOME}/azkaban-web-server/conf/azkaban.properties
      - ./data_web:/azkaban
    ports:
      - "${AZKABAN_WEB_PORT}"
    command: ["sh","-c","/opt/apache/bootstrap.sh web azkaban-azkaban-exec-1 ${AZKABAN_EXEC_PORT}"]
    networks:
      - recommend_network
    healthcheck:
```

```
      test: ["CMD-SHELL", "netstat -tnlp|grep :${AZKABAN_WEB_PORT} || exit 1"]
      interval: 10s
      timeout: 20s
      retries: 3
  azkaban-exec:
    image: moan0/party-api:azkaban-exec-1
    user: "hadoop:hadoop"
    restart: always
    privileged: true
    deploy:
      replicas: ${AZKABAN_EXEC_REPLICAS}
    env_file:
      - .env
    volumes:
      - ./data_exec:/azkaban
      - ./conf/exec/azkaban.properties:${AZKABAN_HOME}/azkaban-exec-server/conf/azkaban.properties
    expose:
      - "${AZKABAN_EXEC_PORT}"
    command: ["sh","-c","/opt/apache/bootstrap.sh exec"]
    networks:
      - recommend_network
    healthcheck:
      test: ["CMD-SHELL", "netstat -tnlp|grep :${AZKABAN_EXEC_PORT} || exit 1"]
      interval: 10s
      timeout: 10s
      retries: 3


#网络配置
networks:
  recommend_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.24.0.0/16


```