一、Docker简介

1. Docker是什么?

产生背景:

- 开发和运维之间因为环境不同而导致的矛盾（不同的操作系统、软件环境、应用配置等）DevOps
- 集群环境下每台服务器都配置相同的环境，太麻烦
- 解决“在我的机器上可以正常工作”的问题

Docker是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

阿里云、百度云等都支持Docker技术

官网：https://www.docker.com/

中文官网：http://www.docker-cn.com/

2. Docker作用

Docker是一种容器技术，使用Docker可以:

- 将软件环境安装并配置好，打包成一个镜像Image，然后将该镜像发布出去（Docker仓库）
- 其他使用者可以在仓库中下载获取这个镜像
- 通过Docker运行这个镜像，就可以获取同样的环境（容器）

Docker简化了环境部署和配置，实现“一次构建，处处运行”，避免了因运行环境不一致而导致的异常

可以将Docker简单的认为是一个虚拟机，可以运行各种软件环境的虚拟机，但与传统虚拟机技术有所不同

Docker容器技术与传统虚拟机技术的区别:

- 传统虚拟机技术：模拟一个完整的操作系统，先虚拟出一套硬件，然后在其上安装操作系统，最后在系统上再运行应用程序

  缺点：资源占用多，启动慢

- Docker容器技术：不是模拟一个完整的操作系统，没有进行硬件虚拟，而是对进程进行隔离，封装成容器，容器内的应用程序是直接使用宿主机的内核，且容器之间是互相隔离的，互不影响

  优点：更轻便、效率高、启动快、秒级

3. 基本术语

术语:

- Docker主机（Host）

安装了Docker程序的主机，运行Docker守护进程

- Docker镜像（Image）

  将软件环境打包好的模板，用来创建容器的，一个镜像可以创建多个容器

- Docker容器（Container）

  运行镜像后生成的实例称为容器，每运行一次镜像就会产生一个容器，容器可以启动、停止或删除

  容器使用是沙箱机制，互相隔离，是独立是安全的

  可以把容器看作是一个简易版的Linux环境，包括用户权限、文件系统和运行的应用等

- Docker仓库（Repository）

  用来保存镜像的，仓库中包含许多镜像，每个镜像都有不同的标签Tag

  官方仓库：https://hub.docker.com/

使用Docker的步骤：

1. 安装Docker
2. 从Docker仓库中下载软件对应的镜像
3. 运行这个镜像，此时会生成一个Docker容器
4. 对容器的启动/停止就是对软件的启动/停止

二、准备Linux系统

1. 安装虚拟机软件

VMware、VirtualBox

Ubuntu、RHEL、CentOS、Debian、SLES（SUSE）

2. 新建虚拟机

版本：Red Hat（64-bit）

网络：桥接网卡

光驱：选择操作系统的iso文件

3. 安装CentOS

4. 连接Linux服务器

步骤：

1. 查看服务器ip地址、

   在虚拟机中执行 `ip addr`

2. 连接服务器

   在客户端中执行 `ssh 服务器账户@服务器地址` ，然后输入密码

   注：SSH是Secure Shell的缩写，用于远程登陆访问的协议

5. 基本操作

5.1 常用命令

```
cat /proc/cpuinfo  # 查看cpu信息
cat /proc/meminfo  # 查看内存信息
uname -r # 查看内核信息，Docker要求CentOS必须是64位，且内核是3.10及以上
sudo reboot # 重启,sudo表示以管理员root身份执行
sudo halt # 关机
```

5.2 安装软件

使用yum，是一个基于rpm的软件包管理工具

用来安装软件包，可以自动解决软件包之间的依赖关系

```
yum install 软件包名  #   安装
yum remove 软件包名 # 卸载
```

三、Docker安装

1. 安装

Docker版本：社区版CE、企业版EE

```
# 设置yum源
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum makecache fast
# 安装Docker-CE
yum -y install docker-ce
```

2. 启动/停止

```
docker version # 查看版本
systemctl start docker #   启动
systemctl stop docker   # 停止
systemctl status docker #   查看状态
systemctl restart docker      # 重启
systemctl enable docker  # 设置开机自动启动
# 验证，运行hello-world
docker run hello-world  # 下载hello-world镜像并运行
```

3. 配置Docker镜像加速

使用阿里云提供的镜像加速（镜像仓库），也可以使用网易云等

步骤：

1. 注册并登陆“阿里云的开发者平台" http://dev.aliyun.com
2. 查看专属的加速器地址
3. 配置自己的Docker加速器

```
vi /etc/docker/daemon.json
  {
    "registry-mirrors": ["https://sswv6yx0.mirror.aliyuncs.com"]
  }
systemctl daemon-reload
systemctl restart docker
```

四、Docker操作

输入 `docker` 可以查看Docker的命令用法，输入 `docker COMMAND --help` 查看指定命令的详细用法

1. 镜像操作

操作	命令	说明
查找	docker search 关键字	可以在Docker Hub网站查看镜像的详细信息，如镜像的tag标签
抽取	docker pull 镜像名:tag	:tag表示软件的版本，如果不指定默认是latest
列表	docker images	查看所有本地镜像
获取元信息	docker inspect 镜像id	获取镜像的元信息，详细信息
删除	docker rmi -f 镜像id或镜像名:tag	删除指定的本地镜像，-f表示强制删除

2. 容器操作

操作	命令	说明
运行	docker run --name 容器名 -i -t -p 主机端口:容器端口 -d -v 主机目录:容器目录:ro 镜像id或镜像名称:tag	--name 指定容器名，名称自定义，如果不指定会自动命名； -i 以交互模式运行，即以交互模式运行容器；-t 分配一个伪终端，即命令行，通常组合使用-it ； -p 指定端口映射，将主机端口映射到容器内的端口；-d 表示后台运行，即守护式运行容器；-v 指定挂载主机目录到容器目录，默认为rw读写模式，ro表示只读
列表	docker ps -a -q	查看正在运行的容器，-a表示显示所有容器，-q表示只显示容器id
启动	docker start 容器id或容器名称	启动容器

停止	docker stop 容器id或容器名称	停止正在运行的容器
删除	docker rm -f 容器id或容器名称	删除容器，-f表示强制删除
日志	docker logs 容器id或容器名称	获取容器的日志
在容器中执行	docker exec -it 容器id或容器名称 /bin/bash	进入正在运行的容器中并开启一个交互模式的终端，可以在容器中执行操作
拷贝文件	docker cp 主机中的文件路径 容器id或容器名称:容器路径；docker cp 容器id或容器名称:容器中的文件路径 主机路径	将文件中的文件拷贝到容器中；将容器中的文件拷贝到主机中
获取元信息	docker inspect 容器id	获取容器的元信息

以CentOS为例：

```
docker search centos
docker pull centos
docker run --name mycentos -it centos:latest  # 根据centos:latest镜像运行容器，并以交互模式进入容器中
# 实际上是在Docker容器中运行一个精简版的CentOS系统
exit # 退出并关闭容器
docker ps -a
docker start mycentos
docker stop mycentos
docker rm mycentos
docker rm -f $(docker ps -aq)  # 强制删除所有容器
```

注：Docker容器内实际上是运行着一个精简版的Linux系统

以tomcat为例：

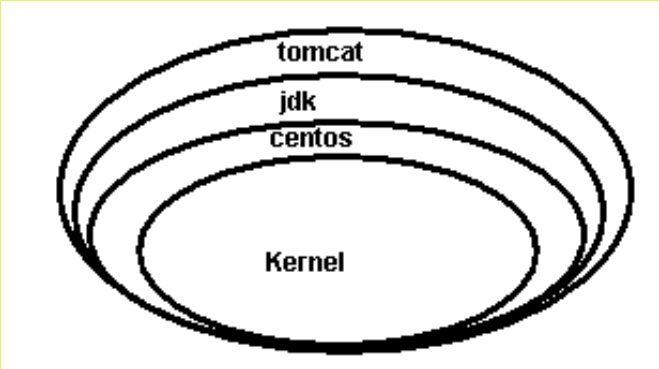
```
# 示例1：基本用法
docker search tomcat
docker pull tomcat
docker run --name mytomcat -p 8888:8080 -d tomcat
# 测试：http://宿主机地址：8888
docker stop mytomcat
docker ps -a
docker start mytomcat

# 示例2：拷贝文件和挂载目录
docker run -p 8080:8080 -d tomcat
docker exec -it 70cba924861c  /bin/bash
cd /usr/local/tomcat/webapps/ROOT
exit
echo welcome to tomcat > index.jsp
docker cp index.jsp 70cba924861c:/usr/local/tomcat/webapps/ROOT  # 将宿主机中的文件拷贝到容器中指定的目录中
#   部署web项目，将war文件放到容器中
docker cp spring-web.war 70cba924861c:/usr/local/tomcat/webapps

# 问题：如果项目更改了需要重新拷贝war文件，太麻烦，可以直接挂载目录（也称为数据卷Volume）
docker run \
-p 8080:8080 \
-v /my/tomcat/webapps/spring-web.war:/usr/local/tomcat/webapps/spring-web.war \
-v /my/tomcat/data:/usr/local/tomcat/dataVolume:ro \
-d tomcat

# 示例3：启动多个容器，一个镜像可以启动多个容器，互相隔离、独立
docker run -p 8081:8080 -d tomcat
docker run -p 8082:8080 -d tomcat
docker run -p 8083:8080 -d tomcat
```

3. 镜像的分层结构

tomcat镜像为什么这么大?

镜像是一种轻量级、可执行的独立软件包，**用来打包软件运行环境和基于运行环境的软件，包含运行某个软件所需要的所有内容。**

基于UnionFS联合文件系统，采用分层结构，一层一层的堆叠起来，像一个同心圆，但从外面来说，只能看到最外层的文件系统（镜像层）



分层结构：共享资源、便于复用（许多镜像都是从相同的Base基础镜像构建而来的，基础镜像只需要保存一份）

镜像都是只读的，而由镜像生成的容器是可修改的

4. 创建镜像

有时从Docker镜像仓库中下载的镜像不能满足我们的要求，此时可以基于这个镜像（基础镜像）封装一个自己的镜像

两种方式：

- 更新镜像：使用docker commit命令
- 构建镜像：使用docker build命令，需要创建Dockerfile文件

4.1 更新镜像

先使用基础镜像创建一个容器，然后对容器进行修改，最后使用commit命令提交为一个新的镜像

步骤：

1. 根据基础镜像，创建容器

```
docker run --name mytomcat -p 8080:8080 -d tomcat
```

2. 修改容器

```
docker exec -it bcd08edac78d  /bin/bash
cd webapps/ROOT
rm -f index.jsp
echo welcome to tomcat > index.html
exit
```

3. 提交为新镜像，语法：`docker commit -m="描述消息" -a="作者" 容器id或容器名 镜像名:tag`

```
docker commit -m="修改默认索引页" -a="汤小洋" bcd08edac78d itany/tomcat:v1
```

4. 使用新镜像运行容器

```
docker run --name tomcat_v1 -p:8080:8080 -d itany/tomcat:v1
```

4.2 构建镜像

根据Dockerfile文件来自动构建镜像

Dockerfile是一个包含创建镜像所有命令的文本文件，使用docker build命令可以根据Dockerfile的内容创建镜像

步骤：

1. 创建一个Dockerfile文件 `vi Dockerfile`

```
# 基础镜像
FROM tomcat

# 作者
MAINTAINER tangxiaoyang@itany.com

# 执行命令
RUN rm -f /usr/local/tomcat/webapps/ROOT/index.jsp
RUN echo "welcome to tomcat!" > /usr/local/tomcat/webapps/ROOT/index.html
```

2. 构建新镜像，语法： `docker build -f Dockerfile文件的路径 -t 镜像名:tag 命令执行的上下文`

```
docker build -f Dockerfile -t itany/tomcat:v2 .
```

3. 使用新镜像运行容器

```
docker run -p 9999:8080 -d itany/tomcat:v2
```

五、Dockerfile详解

1. 简介

Dockerfile是用来构建Docker镜像的文件，是由一系列命令和参数构成的脚本

Dockerfile从FROM命令开始，紧接着各种命令、参数等，最终会生成一个新的镜像

使用Dockerfile构建镜像的步骤：

1. 编写Dockerfile文件
2. 使用docker build构建镜像
3. 使用docker run运行容器

2. 用法

2.1 语法规则

- 指令必须要大写，且后面必须跟参数
- 第一条指令必须是FROM，指定Base Image 基础镜像
- 指令按从上往下的顺序，依次执行
- 每条指定都会创建一个新的镜像层并提交
- `#` 表示注释

2.2 常用指令

指令	解释
FROM	指定基础镜像，即当前新镜像是基于哪个镜像的
MAINTAINER	指定作者
RUN	指定构建过程中要运行的命令
ENV	设置环境变量
WORKDIR	指定默认的工作目录，即进入容器后默认进入的目录
VOLUME	创建挂载点，也称容器数据卷，用于数据共享和持久化
CMD	指定容器启动时要运行的命令，与RUN不同的是，这些命令不是在镜像构建过程中执行的
ENTRYPOINT	指定容器启动时要运行的命令，与CMD有区别
COPY	拷贝文件/目录到镜像中
ADD	拷贝文件到镜像中，且会自动解压缩
EXPOSE	指定对外暴露的端口

3. 案例

3.1 自定义centos镜像

```
# 1.编写Dockerfile文件
vi Dockerfile2
    FROM centos

    MAINTAINER tangxiaoyang@itany.com

    ENV MYPATH /usr/local/centos
    RUN mkdir -p $MYPATH
    WORKDIR $MYPATH

    RUN yum -y install vim

    RUN yum -y install wget

    # 创建挂载点，无法指定宿主机上对应的目录，是自动生成的
    VOLUME  ["/data1","/data2"]
```

```
    CMD ["/bin/bash"]
# 2.使用docker build构建镜像
docker build -f Dockerfile2 -t itany/centos:v1 .
# 3.使用docker run运行容器
docker run -it b25b1dad795c
# 查看镜像的变更历史
docker history b25b1dad795c
# 验证挂载点：
/var/lib/docker/volumes/0b001b4cc8db1ebbbb4c537c17a5c44adb700fb0e1b941bc82cc717c4ae196f6/_data
/var/lib/docker/volumes/f020f5a5664bf68312be9f49a640f27ecfb49990b231aaf3d0eb7cb723fa0ddd/_data
```

CMD和ENTRYPOINT的区别（面试题）：

- CMD

  在Dockerfile中可以有多个CMD指令，但只有最后一条指令生效，所以一般只有一条CMD指令

  CMD指令在被docker run之后的参数覆盖

```
vi aaa
    FROM centos
    CMD ["/bin/ls"]
    CMD ["/bin/bash"]
docker build -f aaa -t itany/aaa .
docker run -it itany/aaa
docker run -it itany/aaa /bin/pwd
```

- ENTRYPOINT

  docker run之后的参数会被作为ENTRYPOINT指令的参数，组合形成新的命令

```
vi bbb
    FROM centos
    ENTRYPOINT ["/bin/ls","/usr/local"]
docker build -f bbb -t itany/bbb .
docker run -it itany/bbb
docker run -it itany/bbb -l  #  ls /usr/local -l
```

3.2 自定义tomcat镜像

```
# 准备工作

# 1.编写Dockerfile文件
vi Dockerfile
```

```
    FROM centos
    MAINTAINER tangxiaoyang@itany.com

    # 拷贝文件，文件必须与Dockerfile在同一目录下
    COPY teacher.txt /usr/local
    ADD jdk-8u171-linux-x64.tar.gz /usr/local
    ADD apache-tomcat-8.5.30.tar.gz /usr/local

    # 配置环境变量
    ENV JAVA_HOME /usr/local/jdk1.8.0_171
    ENV CLASSPATH .:$JAVA_HOME/lib
    ENV CATALINA_HOME /usr/local/apache-tomcat-8.5.30
    ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin

    WORKDIR $CATALINA_HOME

    RUN yum -y install vim

    EXPOSE 8080

    CMD ["catalina.sh", "run"]
# 2.使用docker build构建镜像
docker build -t itany/tomcat:1.0 .
# 3.使用docker run运行容器
docker run \
--name mytomcat \
-p 8080:8080 \
-v /my/tomcat/webapps/spring-web.war:/usr/local/apache-tomcat-8.5.30/webapps/spring-web.war \
-d itany/tomcat:1.0
```

六、使用Docker搭建环境

1. 安装MySQL

```
# 1.拉取镜像
docker pull mysql:5.7
# 2.运行容器
docker run --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -d mysql:5.7
docker exec -it mysql /bin/bash
find / -name "*mysql*"
exit
# 3.创建用于挂载的目录
mkdir -p /my/mysql/conf # 挂载配置文件
mkdir -p /my/mysql/data # 挂载数据文件
mkdir -p /my/mysql/logs # 挂载日志文件
```

```
# 4.拷贝配置文件并修改
docker cp mysql:/etc/mysql/mysql.conf.d/mysqld.cnf /my/mysql/conf/
vi /my/mysql/conf/mysqld.conf
    character-set-server=utf8
# 5.重新运行容器
docker rm -f mysql  # 删除原来的容器
docker run \
--name mysql \
-p 3306:3306 \
-v /my/mysql/conf:/etc/mysql/mysql.conf.d/ \
-v /my/mysql/data:/var/lib/mysql \
-v /my/mysql/logs:/logs \
-e MYSQL_ROOT_PASSWORD=root \
-d mysql:5.7

docker run \
--name mysql57 \
-p 33306:3306 \
--restart always \
-v /my/mysql/conf:/etc/mysql/mysql.conf.d/ \
-v /my/mysql/data:/var/lib/mysql \
-v /my/mysql/logs:/var/log/mysql \
-e MYSQL_ROOT_PASSWORD=p@ssw0rd \
-d mysql:5.7
```

2. 安装Redis

```
#
docker exec -it mysql /bin/bash

mysql -u root -p
docker pull redis
mysql -u root -p -h 宿主机地址
mkdir -p /my/redis/conf
mkdir -p /my/redis/data

# 3.拷贝配置文件并修改
从网上下载一个linux的redis.conf 【http://download.redis.io/releases/下载对应版本的
redis】
wget http://download.redis.io/releases/redis-4.0.10.tar.gz
tar zxf redis-4.0.10.tar.gz
cp redis.conf /my/redis/conf/
vi redis.conf  # bind 127.0.0.1   protected-mode no


    requirepass itany
    appendonly yes
# 4.运行容器
docker run \
--name myredis \
-p 6379:6379 \
-v /my/redis/conf/redis.conf:/usr/local/etc/redis/redis.conf \
-v /my/redis/data:/data \
-d redis redis-server /usr/local/etc/redis/redis.conf
# 5.访问
# 本地访问
docker exec -it myredis /bin/bash
redis-cli
```

```
#   远程访问
使用RedisDesktopManager工具连接
```

3. 安装Nginx

```
# 1.拉取镜像
docker pull nginx
# 2.运行容器
docker run --name mynginx -p 80:80 -d nginx
# 3.创建用于挂载的目录
mkdir -p /my/nginx  # 挂载nginx所有数据
mkdir -p /my/nginx/html # 挂载nginx虚拟主机（网站html数据）
# 4.拷贝配置文件
docker cp mynginx:/etc/nginx/nginx.conf /my/nginx  # 拷贝主配置文件
docker cp mynginx:/etc/nginx/conf.d /my/nginx # 拷贝虚拟主机配置文件
echo welcome to nginx > /my/nginx/html/index.html #   自定义索引页
# 5.重启运行容器
docker rm -f mynginx
docker run \
--name mynginx \
-p 80:80 -p 443:443 \
-v /my/nginx/nginx.conf:/etc/nginx/nginx.conf \
-v /my/nginx/html:/usr/share/nginx/html:ro \
-v /etc/nginx/conf.d:/usr/nginx/conf.d \
-d nginx
# 6.测试
http://宿主机地址
```

七、将本地镜像发布到阿里云

步骤：

1. 登陆“阿里云-开发者平台”，创建命名空间和镜像仓库
2. 将镜像推送到阿里云

```
# 登陆阿里云的docker仓库
docker login --username=tangyang8942@163.com registry.cn-hangzhou.aliyuncs.com
# 创建指定镜像的tag，归入某个仓库
docker tag b25b1dad795c registry.cn-hangzhou.aliyuncs.com/itany/centos:v1.0
# 将镜像推送到仓库中
docker push registry.cn-hangzhou.aliyuncs.com/itany/centos:v1.0
```

3. 拉取镜像

```
docker pull registry.cn-hangzhou.aliyuncs.com/itany/centos:v1.0
```