Docker 课件

一

1. Docker将程序和环境一起打包成镜像,达到应用程式跨平台间的无缝接轨运作。
2. Docker在任何操作系统上都是一样的，而打包的镜像文件在哪个Docker上都可以运行，这就实现了跨平台，跨服务器。只需要一次配置好环境，换到别的机子上就可以一键部署、运行。
3. Docker解决了运行环境和配置问题的软件容器，方便做持续集成并有助于整体发布的容器虚拟化技术。
4. 容器和虚拟机的比较

虚拟机：主操作系统-VMWare-从操作系统-部署应用（资源占用多、冗余步骤多、启动慢）

容器：不是模拟一个完整操作系统而是对进程进行隔离。容器并不需要捆绑一整套操作系统，只需要软件工作所需要的库资源和设置。

原理：

-Docker容器是在操作系统层面上实现虚拟化，直接复用本地主机的操作系统。

-传统虚拟机则是虚拟出一套硬件后，在其上运行一个完整的操作系统，在该系统上再运行所需应用进程。与传统的虚拟机相比Docker优势体现为启动速度快，占用体积小。

1. 镜像（只读模板）-可以创建多个容器-

镜像包括：运行文档

配置环境

运行环境

运行依赖包

操作系统发行版

内核

1. Docker能干嘛？（Build,Ship and Run Any App,Anywhere.）:
2. 更快速的应用交付和部署
3. 更便捷的升级和扩缩容
4. 更简单的系统运维
5. 更高效的系统资源利用
6. Docker是一个C/S模式的架构，通过Docker Client与Docker Daemon建立通信。
7. Docker官网：<http://www.docker.com>

Docker仓库：https://hub.docker.com/

1. 安装Docker

配置环境 yum -y install gcc

yum -y install gcc-c++

安装需要的软件包：yum install -y yum-utils

设置镜像仓（国内阿里云仓库）

yum-cnfig-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

更新yum软件包索引 yum makecache fast

安装docker ce （引擎）：yum -y install docker-ce-cli containerd.io

查看是否安装成功：docker version

启动docker：systemctl start docker

配置阿里云镜像加速器（不然之后下载配置会超时）

1. 为什么Docker比虚拟机快

·Docker有着比虚拟机更少的抽象层

·Docker利用的是宿主机的内核，而不需要再加载操作系统内核

1. 查看虚悬镜像（仓库名，标签都是<none>的镜像，dangling image）
2. 运行容器 docker run [options] imageID [command] [ARG]

options说明：

--name 容器新名字

-d后台运行并返回容器ID，也即启动守护式容器（后台运行）

-i以交互模式运行容器，通常与-t同时使用

-t为容器重新分配一个伪输入终端，通常与-i同时使用也即启动交互式容器（前台有伪终端，等待交互）

-P随机端口映射

-p指定端口映射

docker run -it ubuntu /bin/bash -it是启动交互式终端，/bin/bash是交互式命令接口

1. docker run 的时候会查看本地有没有镜像没有的话先从镜像仓库上拉下来再run一个容器；没有写标签默认是latest
2. 查看正在运行的容器docker ps [options]

-a :列出当前所有正在运行的容器+历史上运行过的

-l :显示最近创建的容器。

-n：显示最近n个创建的容器。

-q :静默模式，只显示容器编号。

1. 两种退出方式 exit 容器停止

ctrl+p+q 容器不停止

1. 有的Docker容器后台运行，就必须有一个前台进程
2. 两种重新进入正在运行的容器的方式：

docker exec -it ID（exit不会停止容器）

docker attach -it ID（exit停止容器）

所以一般用 -d后台启动的程序，再用exec进入对应容器实例

1. 容器文件->电脑磁盘

拷贝：

docker cp 容器ID:容器内路径 目的主机路径

导出容器export：导出整个容器的内容留作为一个tar归档文件

docker export 容器ID > 文件名.tar //导出的tar文件放在当前目录（在目的主机执行导出命令）

导入import：从tar包中的内容创建一个新的文件系统再导入为镜像

cat 文件名.tar | docker import - （镜像用户/)镜像名：版本号

1. Docker镜像加载原理：从底层开始一层一层地加载

bootfs(最底层的引导文件系统)->rootfs（标准目录和文件）

如果多个镜像都从相同的base镜像构建而来，那么只需保存一份base即可

1. 镜像层只读，容器层可写；当容器启动时，一个新的可写层被加载到镜像的顶层，称为“容器层”。下面的镜像层是多个容器共享的。
2. docker commit提交添加功能后的容器使之成为一个新的镜像（镜像-容器-添加内容-容器2.0-commit-镜像2.0)[依然还在本地镜像仓库]

docker commit -m=“提交的描述信息” -a=“作者” 容器ID 目标镜像名：标签

1. 在容器ubuntu上拓展vim：apt-get update apt-get -y install vim
2. 本地镜像发布到阿里云：

aliyun.com->控制台->个人实例->创建命名空间->创建本地仓库->创建后显示如何使用本仓库的命令包括（1）登录阿里云（2）拉取本仓库的镜像（3）上传镜像到本仓库

上传步骤： （1）登录阿里云：

docker login --username=aliyun1214553053 registry.cn-hangzhou.aliyuncs.com

1. 推送到阿里云的仓库

$ docker login --username=aliyun1214553053 registry.cn-hangzhou.aliyuncs.com

$ docker tag [ImageId] registry.cn-hangzhou.aliyuncs.com/baoding2/shangchuanubuntu:[镜像版本号]

$ docker push registry.cn-hangzhou.aliyuncs.com/baoding2/shangchuanubuntu:[镜像版本号]

上传后在此仓库上把镜像拉下来：

$ docker pull registry.cn-hangzhou.aliyuncs.com/baoding2/shangchuanubuntu:[镜像版本号]

1. Dockerhub、阿里云这样的公共镜像仓库可能不太方便，涉及机密的公司不可能提供镜像给公网，所以需要创建一个本地私人仓库供给团队使用，基于公司内部项目构建镜像。

（Docker Registry是官方提供的工具镜像，可用于构建本地私有镜像仓库）

1. 拉取registry镜像：

docker pull registry

1. 运行registry镜像容器：

docker run -d -p 5000:5000 -v /home/baoding/myregistry/:/tem/registry --privileged=true registry

默认情况下，仓库被创建在容器的/var/lib/reegistry目录下，建议自行用容器卷，方便与宿主机联调

1. 查询本地仓库有无镜像：

curl -XGET <http://192.168.111.100:5000/v2/_catalog> (选择ens33的inet)

1. 把镜像打包成符合规范的镜像：

docker tag 16ecd2772934 192.168.111.100:5000/cangkuredis:1.2

1. 修改本地配置文件使之支持http：

vim /etc/docker/daemon.json 添加： ,”insecure-registres”:[192.168.111.100:5000”]

(修改后不生效可以重启docker）

1. 把本地镜像push到本地仓库：

docker push 192.168.111.100:5000/cangkuredis:1.2

1. 从本地仓库上pull镜像：

docker pull 192.168.111.100:5000/cangkuredis:1.2

1. 容器卷：Docker挂载主机目录访问cannot opent directory:Pemission denied

解决办法：在挂载目录后多加一个--privileged=true参数即可（没出错也尽量加上）

docker run -it --priviligede=true -v /宿主机绝对路径目录：/容器内目录 镜像名

特点： 1.数据卷可在容器之间共享或重用数据

1. 卷中的更改可以直接实时生效

（主机和容器的容器卷目录会同时变化，容器停的时候依然会同步修改）

1. 数据卷中的更改不会包含在镜像的更新中
2. 数据卷的生命周期一直持续到没有容器使用它为止

查看是否挂载成功：docker inspect 容器ID (在Mounts里查看）

1. 启动容器时附带容器卷：

docker run -it --name=u5 --privileged=true -v /tmp/dockertmp1:/tmp/docker2tmp ubuntu

docker run -it --name=u5 --privileged=true -v /tmp/dockertmp1:/tmp/docker2tmp:ro ubuntu

ro：read only 在容器内只能读 在主机可以读写

1. 容器2继承容器1的容器卷

docker run -it --privileged=true --volumes-from 父类 --name u2 ubuntu

如：docker run -it --privilegeed=true --volumes-from u5 --name u6 ubuntu

主机和u5容器卷的内容u6也有

1. Docker上安装应用整体步骤：
2. seach 镜像名
3. pull 镜像名：tag
4. docker images 查看镜像
5. docker run -it（-d） -p...... 启动镜像
6. 停止容器
7. 移除容器
8. tomcat运行不起来的解决办法：把容器里的目录/usr/local/tomcat下的webapps文件删除；并把webapps.dist重命名为webapps
9. 启动mysql的命令可在dockerhun官网中查看：<https://hub.docker.com/_/mysql找到>

在docker中运行的mysql容器可在外部用连接软件进行连接

插入中文的时候会报错，需要修改第37集第10min

docker安装mysql并run

出容器后，建议请先修改完字符集编码后再新建mysql数据库-表-插数据

运行的时候一定要挂容器卷

1. 1

DockerFile

1. Dockerfile是用来构建Docker镜像的文本文件，是由一条条构建镜像所需的指令和参数构成 的脚本。
2. DockerFile每条指令都会创建一个新的镜像层并对镜像进行提交
3. FROM：基础镜像，当前新镜像是基于哪个镜像的，指定一个已经存在的镜像作为模板，第一条必须是from
4. MAINTAINER：镜像维护者的姓名和邮箱地址
5. RUN：容器构建时需要的命令（shell或exec）
6. EXPOSE：当前容器对外暴露出的端口
7. WORKDIR：指定在创建容器后，终端默认登陆的进来工作目录，一个落脚点
8. USER：指定该镜像以什么样的用户去执行，如果都不指定，默认是root
9. ENV：用来在构建镜像过程中设置环境变量
10. ADD：将宿主机目录下的文件拷贝进镜像且会自动处理URL和解压tar压缩包
11. COPY：类似ADD，拷贝文件和目录到镜像中。将从构建上下文目录中 <源路径> 的文件/目录复制到新的一层的镜像内的 <目标路径> 位置。COPY src dest 、 COPY ["src", "dest"] 、 <src源路径>：源文件或者源目录 、 <dest目标路径>：容器内的指定路径，该路径不用事先建好，路径不存在的话，会自动创建。
12. VOLUME：容器数据卷，用于数据保存和持久化工作
13. CMD：指定容器启动后的要干的事情。Dockerfile 中可以有多个 CMD 指令，但只有最后一个生效，CMD 会被 docker run 之后的参数替换
14. ENTRYPOINT：类似于 CMD 指令，但是ENTRYPOINT不会被docker run后面的命令覆盖，而且这些命令行参数会被当作参数送给 ENTRYPOINT 指令指定的程序（如果 Dockerfile 中如果存在多个 ENTRYPOINT 指令，仅最后一个生效。）
15. 写好Dockerfile后需执行docker build -t centosjava8:1.5 . 形成一个新镜像（.是当前目录，在其之前有一个空格）
16. 查看所有的虚悬镜像：

docker images ls -f dangling=true

移除所有的虚悬镜像：

docker image prune

1. 微服务打包成jar包->在同一目录下创建Dockerfile->docker build形成镜像

Docker Compose ( 负责对Docker容器集群的快速编排)

1. Docker Compose可以管理多个Docker容器组成一个应用。配置docker-compose.yml文件，写好多个容器之间的调用关系。然后一个命令就能同时启动或关闭这些容器。（如一个项目需要用到多个软件）
2. Docker Compose使用步骤：
   1. 编写Dockerfile定义各个微服务应用并构建镜像
   2. 配置docker-compose定义完整的业务单元 //vim
   3. -compose.yml
   4. 执行docker-compose up命令，完成一件部署上线（等价于一次运行多个docker run）
3. docker-compose.yml例子

|  |
| --- |
| version: '3'  services:    mysql:      image: mysql:5.7      command: --default-authentication-plugin=mysql\_native\_password      container\_name: mysql      hostname: mysqlServiceHost      network\_mode: bridge      ports:      - "3306:3306"      #restart: always      restart: on-failure      volumes:      - ./mysql:/var/lib/mysql      - ./my.cnf:/etc/mysql/conf.d/my.cnf      - ./mysql/init:/docker-entrypoint-initdb.d/      - ./shop.sql:/docker-entrypoint-initdb.d/shop.sql      environment:      - MYSQL\_ROOT\_PASSWORD=a123456      - MYSQL\_USER=root      - MYSQL\_PASSWORD=a123456      - MYSQL\_DATABASE=shop    redis:      image: redis:3      container\_name: redis      hostname: redisServiceHost      network\_mode: bridge      restart: on-failure      ports:      - "6379:6379"    golang:      build: .      restart: on-failure      network\_mode: bridge      ports:      - "8080:8080"      links:      - mysql      - redis      volumes:      - /Users/mac/go/src/gitee.com/shirdonl/LeastMall:/go      tty: true |