Docker

1. Docker简介

a) 出现的背景：一款产品从开发到上线，从OS到运行环境，再到应用配置。开发与运维之间的协作不是很方便，特别是各种版本的迭代之后，不同版本环境的兼容，对运维人员都是考验。

b) Docker提供了一个标准化的解决方案。--- **软件可以带环境安装，安装的时候，把原始环境一模一样地复制过来。以此来消除协作编码时的一些问题。**

c) 镜像技术：从系统环境开始，自底向上打包应用

i) OS环境 → 运行依赖包 → 运行环境 → 配置环境 → 运行文档

ii) 由此达到应用跨平台间的无缝接轨运作。类似JVM的概念

定义：

- 解决了运行环境和配置问题软件容器（Container），方便做持续的集成并有助于整体发布的**容器虚拟化技术**

2. 与VM的区别

1) 虚拟机(Virtual Machine)

就是带环境安装的一种解决方案。

但是有些缺点：

- **启动速度慢（分钟级） -- docker milliseconds**
- **资源占用多**

2) Linux Containers (LXC)

Linux容器不是模拟一个完整的操作系统，而是对进程进行隔离

3) DevOps 开发/运维 工程师 --- 一次构建 → 随处运行

3. Docker三要素

1) **Image**

镜像就是一个**只读**的模板。Image可以用来创建Container， 一个镜像可以创建很多容器。

2) **Container**

Docker利用Container独立**运行的一个/一组应用**。

容器是用镜像创建的运行实例，可以被启动、开始、停止、删除。

每个容器是相互隔离的、保证安全的平台。

可以把容器看做是一个简易版的Linux环境

3) **Repository**

存放Image的地方

- 不同于Registry，Registry上往往存放着多个Repository，每个Repo可以有多个镜像

4. 运行底层原理

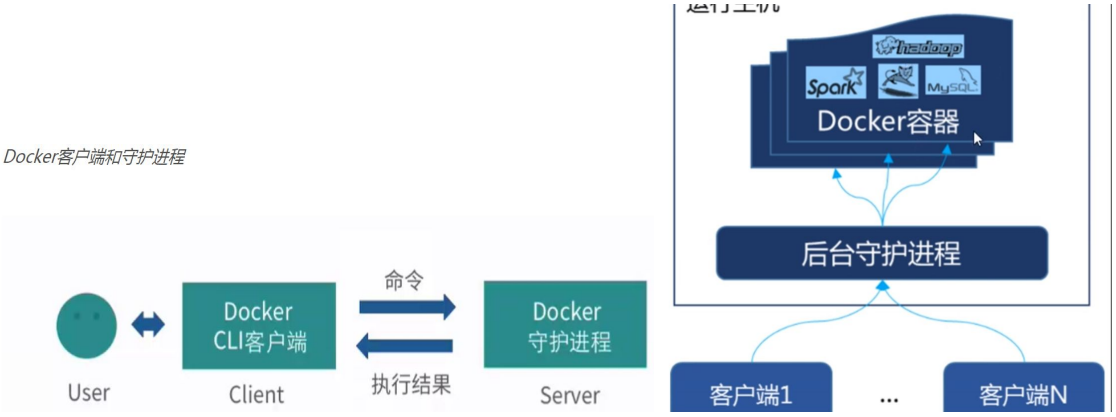
1) Docker是怎么工作的

a) Client-Server结构的系统

b) Docker守护进程运行在主机上，然后通过Socket连接从客户端访问

c) 守护进程从客户端接受命令并管理运行在主机上的容器

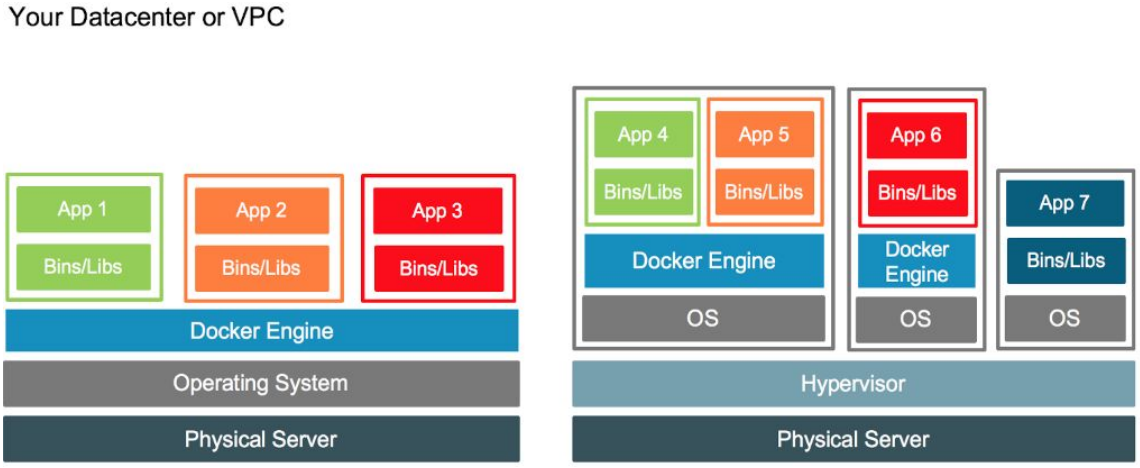
Docker客户端和守护进程



2) **为什么更快?**

a) **有更少的抽象层，不需要Hyperisor实现硬件资源虚拟化**

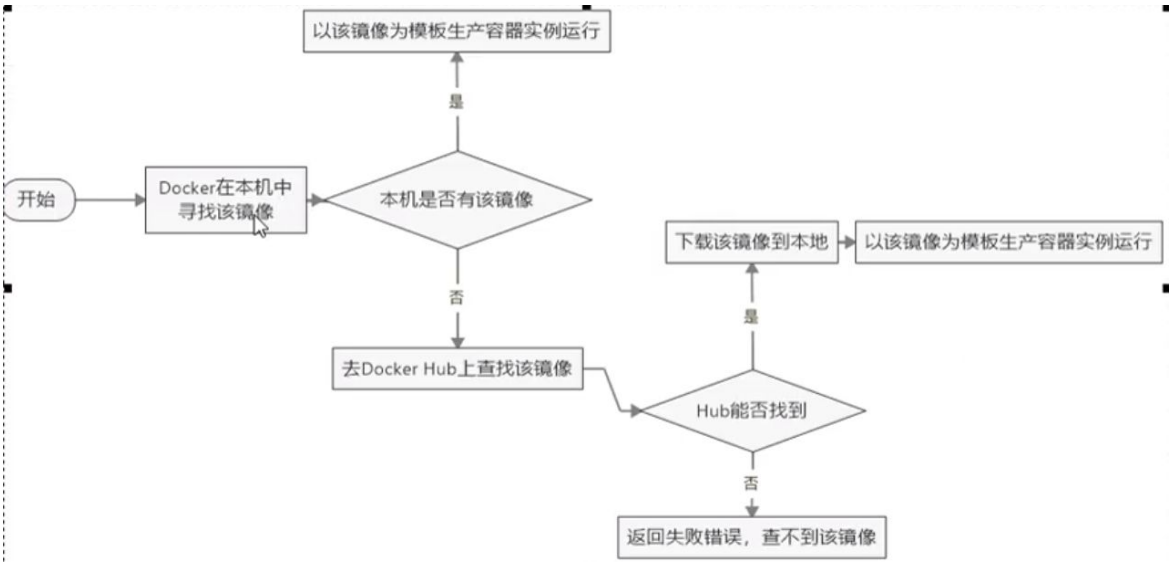
b) **利用宿主机内核，不需要加载Guest OS**



- A hypervisor (or virtual machine monitor, VMM) is a computer software, firmware or hardware that creates and runs virtual machines.

5. 操作 (MacOS下)

1) Docker run hello-world
   - 先是会在本地找有没有该镜像、如果没有则会从dockerhub拉下来
2) 具体流程如下



3) **相关命令行**

- **帮助命令**
  - Docker - verison, info, help
- **镜像命令**
  - **Docker images** --- 列出本地主机上的镜像
    - Repository
    - TAG: 代表Repo的不同版本
    - **-a: 列出本地所有的镜像(含中间映像层)**
    - **-q: 只显示镜像ID**
  - Docker search
  - Docker pull
  - Docker rml --- 删除镜像

- **容器命令**
  - **1) 新建并启动容器**
    - docker run [OPTIONS] IMAGE [COMMAND] [ARG..]

    - OPTIONS：
      - **--name** = "容器的新名字"：为容器指定一个名称
      - **-d:** 后台运行容器，并返回容器ID，也即启动守护式容器
      - **-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**
      - -P: 随机端口映射
      - **-p: 指定端口映射，有四种格式：**
        - Ip:hostPort:containerPort
        - ip::containerPort
        - **hostPort:containerPort**
        - containerPort

- **2) 列出当前所有正在运行的容器**
  - docker ps [OPTIONS]

  - 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
    - --no-trunc：不截断输出

- **3) 退出容器**
  - exit：容器停止退出
  - ctrl+P+Q：容器不停止退出
- **4) 启动/重启容器**
  - docker start　容器ID / 容器名
  - docker restart　容器ID / 容器名
- **5) 停止/强制停止容器**
  - docker stop　容器ID / 容器名
  - docker kill　容器ID / 容器名
- **6) 删除已经停止的容器**
  - docker rm 容器ID
  - 一次性删除多个容器
    - docker rm -f (docker ps -a -q)
    - docker ps -a -q | xargs docker rm

- **重要命令**
  - **1）启动守护式容器**
    - docker run -d 容器名
    - docker run -d ubuntu → 用docker ps -a 进行查看，会发现容器已经退出
    - Docker容器后台运行，就必须有一个前台进程，如果不是那种一直挂起的命令，就会自动退出
  - **2) 查看容器日志**
    - docker logs -f -t --tail 容器ID
      - -t 是加入时间戳
      - -f 跟随最新的日志打印
      - --tail 数字 - 显示最后多少条
  - docker run -d ubuntu /bin/sh -c “while true;do echo hello zzyy;sleep 2;done”
  - **3) 查看容器内运行的进程**
    - docker top 容器ID
  - **4) 查看容器内部细节**
    - docker inspect 容器ID
    - 以JSON串的形式打印
  - **5) 进入正在运行的容器内并以命令行交互**
    - docker exec -it 容器ID bashShell
    - 重新进入docker attach 容器ID

Attach: 直接进入容器启动命令的终端，不会启动新的进程
Exec： 是在容器中打开新的终端，并可以启动新的进程

- **6) 从容器内拷贝文件到主机上**
  - docker cp 容器ID：容器内路径 目的主机路径

面试题：Docker是如何实现各个容器之间分离的？

- **控制组（cgroups）**是 Linux 内核的一个特性，主要用来对共享资源进行隔离、限制、审计等。只有能控制分配到容器的资源，才能避免当多个容器同时运行时的对系统资源的竞争。

CAP

ACID

6. Docker镜像

6.1 是什么

a) **UnionFS -- Union File System**
   i) 支持对文件系统的修改作为一次提交来一层层叠加
   ii) 一次同时加载多个文件系统，但从外面看来只能看到一个文件系统

b) **Docker镜像加载原理**
   i) docker的镜像实际上是由一层层的文件系统组成，UnionFS
   ii) Boofs 主要包含 bootloader & kernel -- 最底层实现
   iii) boofs → rootfs（/dev，/bin）--> ubuntu / centOS

c) **分层的镜像**
   i) Tomcat 为例：kernel+os+jdk8+tomcat

d) **为什么Docker镜像要采用这种分层结构呢？**
   i) **共享资源**
   比如A & B都需要base镜像，那么宿主机只需要在磁盘上保存一份base镜像
   同时内存中也只需加载一份base镜像，而后为A & B 服务

6.2 特点

- Docker镜像都是只读的
- 当容器启动时，一个新的可写层被加载到镜像的顶部
- 这一层通常被称作“容器层”，以下都是“镜像层”

6.3 Docker镜像commit操作补充

- Docker commit 提交容器副本使其成为一个新的镜像
- 举个例子的话就是: 从hub拉下一个image，改成自己想要的，然后又commit成一个自定义的新的镜像
- **docker commit -a=“author” -m=”msg” ID**
  - docker run -it -p 8080:8080 tomcat
    - -p: 主机端口：docker容器端口
    - -P: 随机分配端口
    - -i: 交互
    - -t: 终端
  - docker run -it -p 5984:5984 -e COUCHDB_USER=admin -e COUCHDB_PASSWORD=admin couchdb

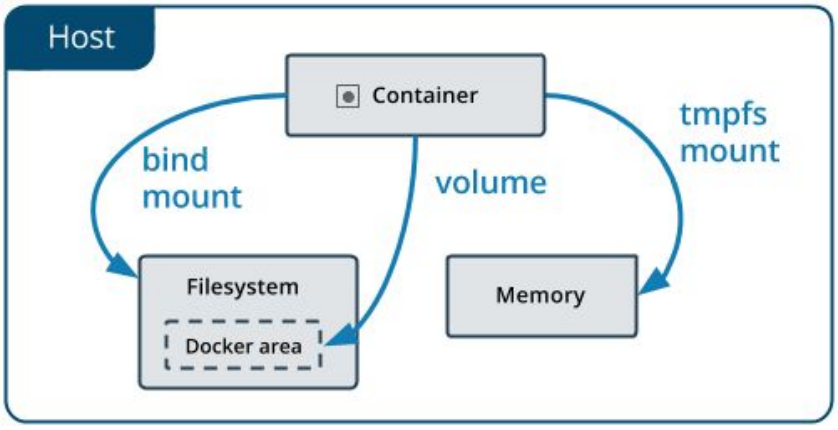
7. 容器数据卷

7.1 容器数据的持久化 --- volume

- 卷的设计目的就是数据的持久化，完全独立于容器的生存周期。因此Docker不会在容器删除时删除其挂载的数据卷。

7.2 数据卷的特点

1. 数据卷可在容器之间共享或重用数据
2. 圈中的更改可以直接生效
3. 数据卷中的更改不会包含在镜像的更新中
4. 数据卷的生命周期一直持续到没有容器使用它为止



7.3 数据卷

容器内添加

1. 直接命令添加
    a. docker run -it -v /宿主机绝对路径目录：/容器内目录　镜像名
    b. 查看数据卷是否挂载成功
    c. 容器和宿主机之间**数据共享**
    d. 容器停止退出后，主机修改后数据是否同步 -- **restart后同步了**
    e. 命令（带权限）
        i. **docker run -it -v host:container:ro imageName**
        ii. **Ro → read only**
        iii. **容器还是能够收到文件，但是无法修改**
2. DockerFile添加
    a. 根目录下新建mydocker文件夹并进入
    b. 可在Dockerfile中使用**VOLUME指令来给镜像添加Data Volume**
    c. File构建
    d. Build后生成镜像
    e. run 容器
3. 备注

8. DockerFile

8.1 流程

1) 写一个dockerfile文件
2) docker build 之后获得一个自定义的镜像
3) run image

8.2 是什么

- dockerfile是用来构建Docker镜像的构建文件，是由一系列命令和参数构成的脚本

8.3 DockerFile构建过程解析

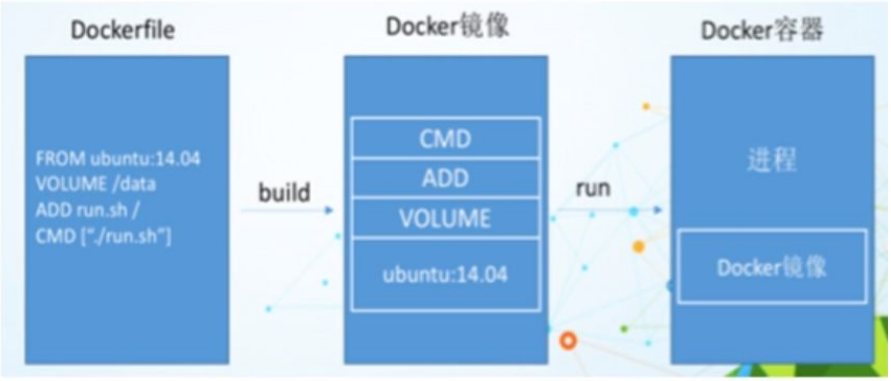
1. 每条保留字指令都必须为大写字符且后面要跟随至少一个参数
2. 指令按照从上到下，顺序执行
3. # 表示注释
4. 每条指令都会创建一个新的镜像层，并对镜像进行提交

8.4 Docker执行DockerFile的大致流程

1. Docker从基础镜像运行一个容器 --- FROM
2. 执行一条指令并对容器做出修改
3. 执行类似docker commit的操作提交一个新的镜像层
4. Docker再基于刚提交的image运行一个新的容器
5. 执行dockerfile中的下一条指令直到所有指令都执行完成

8.5 小总结

Dockerfile -- 软件的原材料



DockerFile 体系结构 -- 保留字指令

Keywords	Meaning
FROM	基础镜像
MAINTAINER	镜像维护者的name & email
RUN	容器构建是需要运行的命令
EXPOSE	当前容器对外暴露出的端口
WORKDIR	制定在创建容器后，终端默认登陆的进来工作目录
ENV	在构建镜像过程中 - 设置环境变量
ADD	拷贝+解压缩
COPY	拷贝
VOLUME	容器数据卷，用于数据保存和持久化工作
CMD	指定一个容器启动时要运行的命令，DockerFile中可以有多个命令， 但是只有最后一个会生效，CMD会被docker run之后的参数替换
ENTRYPOINT	== CMD，但是不会被替换，会追加
ONBUILD	当构建一个被继承的Dockerfile时运行，父镜像在被子继承后父镜像的onbuild被触发

附：Linux 常用命令

1. ps - ProcesseS

- Linux ps命令用于显示当前进程 (process) 的状态。
- **ps [options] [--help]**
- ps -ef //显示所有命令，连带命令行

2. top - Top Processes

- Linux top命令用于实时显示 process 的动态。
  top命令会默认按照CPU的占用情况，显示占用量较大的进程,可以使用top -u 查看某个用户的CPU使用排名情况。

3. ls

- List
- 列出当前工作目录的内容

4. mkdir

- Make Directory
- 用于新建一个新目录

5. cd

- Change Directory
- 切换文件路径，cd 将给定的文件夹（或目录）设置成当前工作目录

6.rmdir

- Remove Directory
- **删除给定的目录**

7.rm

- Remove
- 删除给定的文件

8.cp

- Copy
- 命令对文件进行复制

9. mv— Move

10. cat— concatenate and print files

11. tail — print TAIL(from last)

12. grep

- 在给定的文件中搜寻指定的字符串

13. .su — Switch User