Docker 学习笔记

> Author：Marlous
>
> E-mail：goonecat@foxmail.com

> Date：2018/10/23

> 参考文章

一 Docker 概念与作用

“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，容器是完全使用沙箱机制，相互之间不会有任何接口。”

开发、测试、上线（运维）过程中，将环境打包到容器中，方便部署，开箱即用； 可以直接使用现成的容器（打包好的环境）； 可以制作自己的容器。

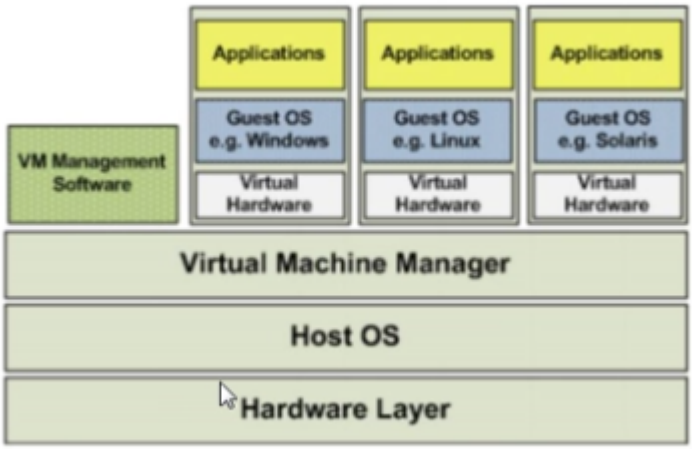
二 安装运行

1. 去 Docker 官网下载安装包安装。下载前需要注册，因为注册用的 Google 人机验证，所以可能无法显示点击 sign up 按钮。
2. Docker for Windows Installer（需要 Windows 10 专业版本或企业版，具体见官网。）使用的是微软的 hyper-V，由于 Hyper-V 与 VirtualBox 冲突导致安装上无法使用的情况（如果安装过 virtualbox，32 位的可以和 Hyper-V 共存），可换为 DockerToolbox，后者利用 virtualbox。
3. Tips：
   点击 Docker Quickstart Terminal，如果先安装过 git，会出现 “windows 正在查找 bash.exe” 问题，是因为快捷方式路径不对，因为需要 Git bash.exe 来执行 docker star.sh（git bash 是 Windows 下的命令行工具，基于 msys GNU 环境）。
   快捷方式目标的正确路径书写示例：`"D:\Program Files\Git\bin\bash.exe" --login -i "C:\Program Files\Docker Toolbox\start.sh"`

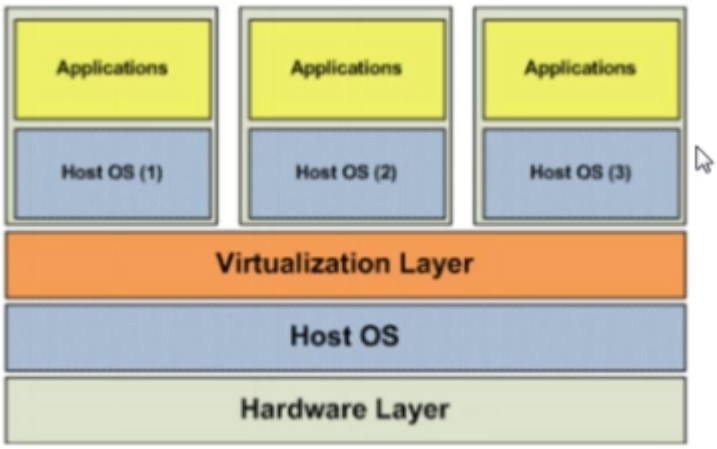
三 Docker

1 虚拟化种类

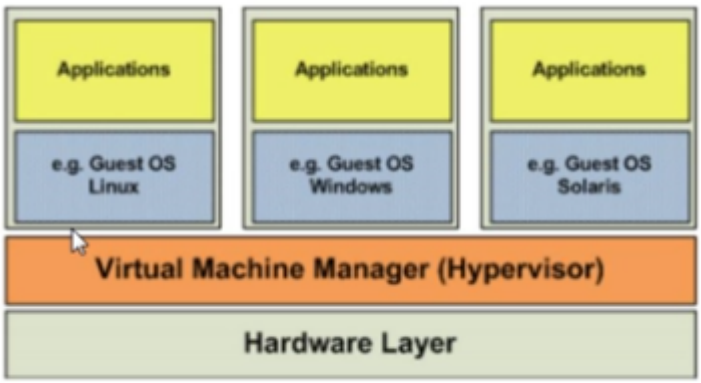
- 全虚拟化：应用程序实现。



- OS 层虚拟化：



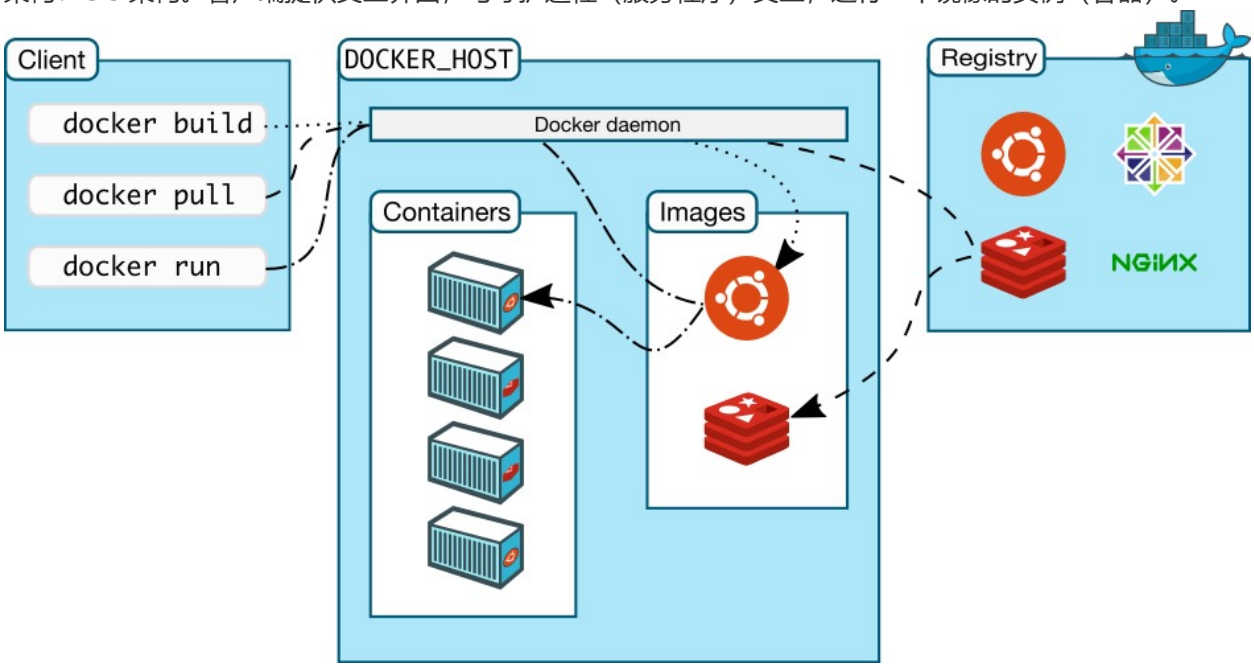
- 硬件层虚拟化：



2 Docker 原理与组成

1. 组成：Docker Client 客户端、Docker Daemon 守护进程、Docker Image 镜像、Docker Container 容器。
2. 原理：容器技术类似 OS 层虚拟化（不等同），容器本质上是基于应用的，虚拟出运行环境。容器相当于一个个运行的小操作系统，但它是共享宿主机操作系统内核的。

3. 架构：C/S 架构。客户端提供交互界面，与守护进程（服务程序）交互，运行一个镜像的实例（容器）。



四 Docker 快速上手

参考资料

1 基本命令

1. 镜像操作：

```
# 列出镜像
docker images

# 搜索镜像
docker search

# 拉取镜像，docker hub 中拉取（下载）
docker pull

# 删除镜像
docker rmi 镜像ID
# 删除全部镜像
docker rmi `docker images -q`
```

2. 容器操作：

```
# 查看正在运行的容器
docker ps
-a    # 参数查看全部
-l    # 参数查看最后一次运行的
-f status=exited    # 查看停止的
```

3. 创建与启动容器：

```
docker run
-i   # 运行容器
-t   # 启动后进入命令行
-it
--name=名称    # 为创建的容器命名
-d    # 后台启动（启动后不进去）。
-v    # 目录映射，前者为宿主机目录，后者为容器目录。通过宿主机操作容器，--v 可做多个映射。
-p    # 端口映射，前者为宿主机端口，后者为容器端口。 --p 做多个端口映射。

# 退出容器
exit

--------------------------------------------------------------------------
交互式容器启动：进容器运行，退出停止运行。
docker run -it --name=起个容器名 镜像名:标签 [命令解释器路径]

守护式容器启动：
docker run -id --name=起个容器名 镜像名:标签
docker exec -it 容器名    # 进入守护式容器

--------------------------------------------------------------------------
docker start  # 运行容器
docker stop   # 停止容器
```

4. 文件拷贝：

```
docker cp 源文件目录 容器名:容器目录
docker cp 容器名:容器目录 目标文件目录
```

5. 目录挂载：

```
docker run -di --name=容器名 -v 宿主机目录路径:容器内目录路径 镜像名称:标签
--priviledged=true  # 获取特权，访问共享的目录
```

6. 查看 IP 地址与删除容器

```
docker inspect 容器名

docker rm 容器名
```

2 制作、备份与迁移

1. 制作镜像：

- 将容器制作为镜像：

```
docker commit 容器名 起个镜像名
```

- 用 dockerfile 构建一个镜像：类似于安装脚本。
  选一个空目录；编写 dockerfile；　在 Dockerfile 文件所在目录执行构建命令。
  参考博文 1 参考博文 2

2. 镜像备份：
   备份及恢复（导出导入）的镜像都在宿主机执行 docker 命令所在位置。

```
docker save -o 压缩包名.tar 镜像名
```

3. 镜像恢复：
   备份及恢复（导出导入）的镜像都在宿主机执行 docker 命令所在位置。

```
docker load -i 压缩包名.tar
```

3 部署运行实例

实例：部署运行 MySql

1. 获取镜像：

- 通过 pull 镜像：

```
docker search mysql
docker pull mysql:5.6
docker images |grep mysql
```

- 使用容器保存的镜像。
- 通过 Dockerfile 构建镜像：参考文章
  用 Dockerfile 脚本做镜像的定制，实际上就是定制每一层所添加的配置、文件。

```
进入创建的 mysql 目录，创建 Dockerfile
编写脚本。
docker build -t mysql

------------------------------------------------------------
# 先创建准备存放文件的文件夹
mkdir -p ~/mysql/data ~/mysql/logs ~/mysql/conf

data 目录将映射为 mysql 容器配置的数据文件存放路径
logs 目录将映射为 mysql 容器的日志目录
conf 目录里的配置文件将映射为 mysql 容器的配置文件

然后使用制作好的镜像。
```

2. 使用镜像：

```
docker run -p 3306:3306 --name [mymysql] -v [$PWD/conf:/etc/mysql/conf.d] -v [$PWD/logs:/logs] -
v [$PWD/data:/var/lib/mysql] -e [MYSQL_ROOT_PASSWORD=123456] -d [mysql:5.6]

# 参数
-p 3306:3306：将容器的 3306 端口映射到主机的 3306 端口。
-v -v $PWD/conf:/etc/mysql/conf.d：将主机当前目录下的 conf/my.cnf 挂载到容器的 /etc/mysql/my.cnf。
-v $PWD/logs:/logs：将主机当前目录下的 logs 目录挂载到容器的 /logs。
-v $PWD/data:/var/lib/mysql ：将主机当前目录下的data目录挂载到容器的 /var/lib/mysql 。
-e MYSQL_ROOT_PASSWORD=123456：初始化 root 用户的密码。
```

五 补充

> Dockerfile 构建：参考博文

团队采用两个 Dockerfile，一个负责开发环境的镜像构建，一个负责生产环境的镜像构建。 开发镜像包含了代码构建所需要的环境，镜像大小自然比较大，生产镜像仅包含应用运行所需要的内容，是很精简的体积很小的镜像。