Docker

主讲：军 哥

蛙课网（动力节点旗下品牌）

http://www.wkcto.com

第一章 Docker 概述

1-1 虚拟化技术发展史

在虚拟化技术出现之前，如果我们想搭建一台服务器，我们需要做如下的工作：

→ 购买一台硬件服务器；

→ 在硬件服务器上安装配置操作系统系统；

→ 在操作系统之上配置应用运行环境；

→ 部署并运行应用；

这种方式的缺点就是：

→ 部署应用非常慢；

→ 需要花费的成本非常高（时间成本、服务器成本）；

→ 应用迁移麻烦；要将应用迁移，又得重复部署应用的过程：购买服务器 -> 安装操作系统 OS -> 配置运行环境 -> 部署应用

所以，为了解决这个问题，后续出现了虚拟化技术。

1-2 虚拟化技术是什么?

虚拟化（英语:Virtualization）是一种计算机资源管理技术，是将计算机的各种硬件资源，比如服务器、网络、CPU、内存及存储等，予以抽象和转换后呈现出一套新的硬件资源环境，在这一套新的硬件环境下可以安装我们的操作系统，部署我们的应用运行环境等，它打破计

算机硬件资源不可切割的障碍，使我们可以比原本的计算机硬件资源结构更好的方式来组合应用这些资源。

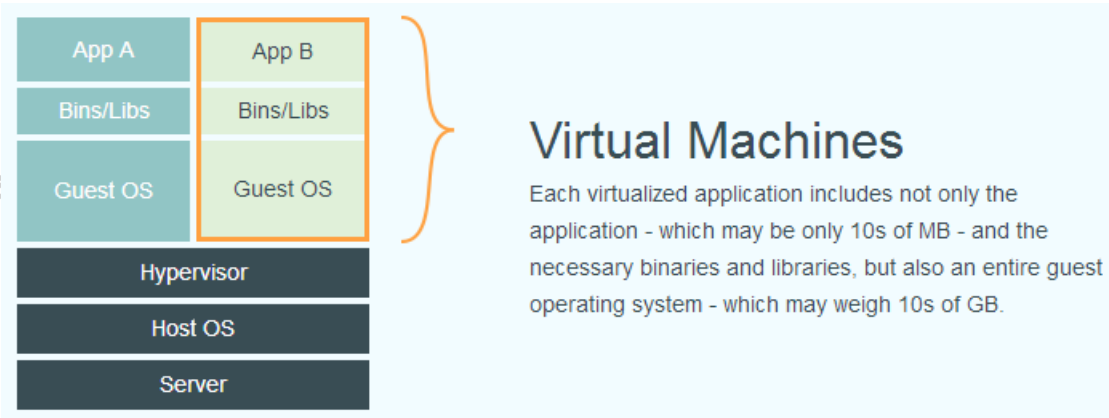
1-3 虚拟化技术的分类

虚拟化一般分为：

硬件级虚拟化（hardware-level-virtualization）

操作系统级虚拟化（os-level-virtualization）

硬件级虚拟化 是运行在硬件之上的虚拟化技术，它的核心技术是 Hypervisor 发音 [,haipə'vaizə]，Hypervisor 是一种运行在基础物理服务器硬件之上的软件层，可以虚拟化硬件资源，例如 cpu、硬盘、内存资源等。然后我们可以通过在虚拟化出来的资源之上安装操作系统，这也就是所谓的虚拟机。像 VMWare，VirtualBox 等都是使用该技术，我们经常使用的桌面版的虚拟机 VMWare 就是采用这种虚拟化技术。如下图所示：



通过 Hypervisor 层，我们可以创建不同的虚拟机，并且每个虚拟机都

是分离、独立的，这样一来，我们就可以在一台硬件服务器和本地操作系统之上虚拟化出多个服务器，用来部署我们的应用；

1-4 虚拟化技术的优缺点

虚拟化技术的优点：

一台物理服务器可以虚拟化出多个虚拟的服务器，让计算机资源得以充分利用；

虚拟化技术的缺点：

1、每创建一个虚拟机的时候，都会创建一个操作系统，这个操作系统会占用很多资源，这样无疑大大的增加了资源的消耗，当安装的虚拟机越多，资源消耗就越多。

2、环境兼容性问题，开发时的环境运行正常，部署到虚拟机环境测试则有可能发生错误；

1-5 容器技术的发展

基于硬件级虚拟化技术的缺点和不足，后续又发展出来了另一种虚拟化技术，即操作系统级虚拟化技术；

操作系统级虚拟化 是运行在操作系统之上的虚拟化技术，它模拟的是运行在一个操作系统上的多个不同进程，并将其封装在一个密闭的容器里面，该技术也称为容器化技术。

在容器化技术领域，Docker 是目前最流行的一种实现。Docker 发布

于 2013 年，Docker 基于 LXC 技术，LXC 是 Linux 平台上的容器化技术实现。

注：LXC 是 Linux Container 的简写，它是一种内核虚拟化技术，可以提供轻量级的虚拟化，以便隔离进程和资源，它与宿主机使用同一个内核，性能损耗小，这种技术是 Linux 提供的，但是直到 Docker 出世，该技术才被发挥出来。

1-6 Docker 的发展历史

2010年，几个年轻人在旧金山成立了一家做PaaS（Platform as a Service，平台及服务）平台的创业公司，起名为 dotCloud，并且还获得了创业孵化器 Y Combinator 的支持，虽然 dotCloud 期间获得过一些融资，但随着 IT 巨头（微软、谷歌、亚马逊等）也杀入 PaaS 平台，dotCloud 举步维艰。

2013 年，dotCloud 的创始人，28 岁的 Solomon Hykes 做了一个艰难的决定：将 dotCloud 的核心引擎开源，这项核心引擎技术能够将 Linux 容器中的应用代码打包，轻松的在服务器之间迁移。

然而这个基于 LXC（Linux Container）技术的核心管理引擎开源后，让全世界的技术人员感到惊艳，感叹这一切太方便了……。也正是 dotCloud 的创始人这个艰难的孤注一掷的决定让所有的 IT 巨头们也为之一颤。

Docker 创始人 Solomon Hykes （Docker 之父）

从 2013 年 Docker 开源开始，Docker 技术风靡全球，于是 dotCloud 公司决定将 Docker 作为主要业务进行发展，并把公司改名为 Docker Inc，全身心投入到 Docker 的开发中，并于 2014 年 8 月，Docker 宣布把 PaaS（Platform as a Service，平台及服务）业务 dotCloud 出售给位于德国柏林的同样专注于平台即服务业务的提供商 cloudControl，从此 Docker 可以轻装上阵，专注于 Docker 的研发。

从 2013 年 2 月决定开源，到 2013 年 3 月 20 日发布 Docker 0.1，只用了一个月时间。当前 Docker 的最新版本是 18.03；

Docker 迅速成长，在 2014 年 6 月 9 日，Docke 团队宣布发布 Docker 1.0，1.0 版本标志着 Docker 平台已经足够成熟，并可以被应用到生产产品中（还提供了一些需要付费的支持选项）。

一年的时间，使一个围绕着 Docker 的小型初创企业生态体系逐渐形成。Docker 先后赢得了 Google、 微软、Amazon、VMware 等 IT 巨头的青睐，他们纷纷表示将保证自己平台与 Docker 容器技术的兼容性。

2016 年 2 月 29 日，CloudControl 公司在其官方博客中宣告即将破产，隶属于 cloudControl 公司的 dotCloud 也宣布将于 2 月 29 日关闭服务。作为 Docker 的前身，DotCloud 目睹 Docker 的成长，成为云平台的一颗新星，而自己却力不从心，Docker 的繁荣间接地导致 Docker 的前身 dotCloud 在 PaaS 平台的衰败，兴衰成败，令人唏嘘不已，这也许是颠覆式创新的经典案例。

1-7 Docker 是什么

1、Docker 是一个开源的应用容器引擎，它基于 Google 公司推出的 Go 语言实现，项目代码托管在 GitHub 上进行维护；

https://github.com/docker/docker-ce

2、Docker 技术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，打包好的容器可以发布到任何流行的 Linux 服务器上运行，这样就可以解决开发环境与运维环境不一致的问题了，所以容器技术解决了开发和运维之间的矛盾，让开发专注于开发，运维专注于运维，不要被环境问题所打扰；

3、Docker 彻底释放了虚拟化的威力，极大降低了计算机资源供应的

成本，Docker 重新定义了程序开发测试、交付和部署过程，Docker 提出了“构建一次，到处运行”的理念，让应用的开发、测试、部署和分发都变得前所未有的高效和轻松！

4、Docker 是一种轻量级的操作系统虚拟化解决方案，Docker 的基础是 Linux 容器（LXC）技术，在 LXC 的基础上 Docker 进行了进一步的封装，让用户不需要去关心容器的管理，使得操作更为简便。用户操作 Docker 的容器就像操作一个快速轻量级的虚拟机一样简单；

Docker 自开源后受到广泛的关注，Docker 最早是基于 Ubuntu 开发的，但后续 CentOS、Debian、Fedora 等主流的 Linux 操作系统都支持 Docker；

总结：简单地说，Docker 是对软件和其依赖环境的标准化打包，应用之间相互隔离，共享一个 OS Kernel（解决了资源浪费的问题），可以运行在很多主流操作系统上；

但是也需要澄清一下，Docker 本身不是容器，Docker 只是管理容器的引擎。

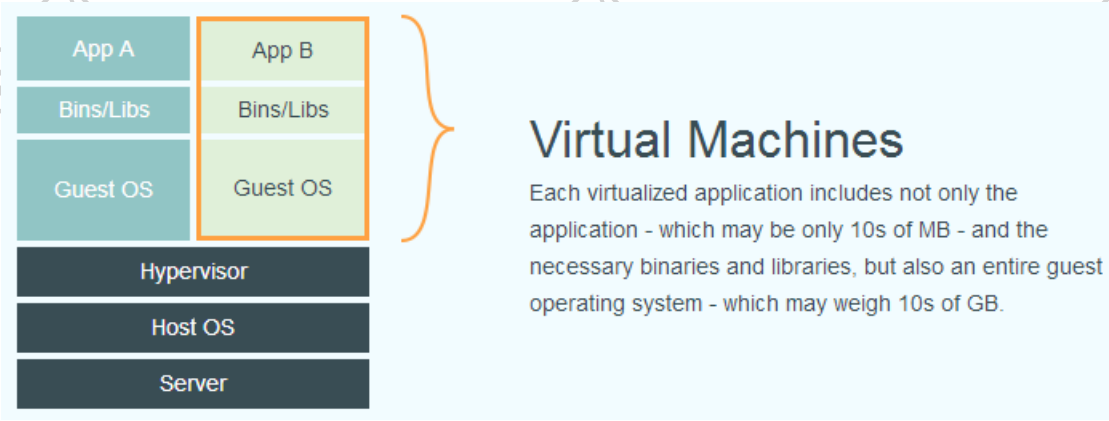
1-8 容器和虚拟机的区别

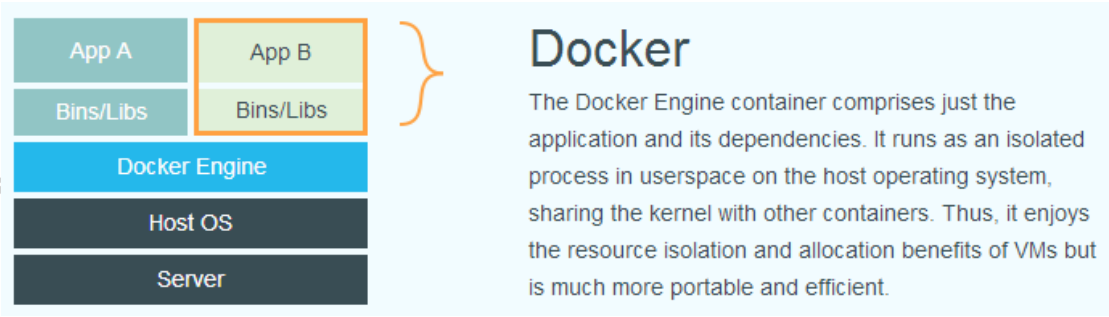
容器是将代码和环境的关系打包在一起的一个集合，而虚拟机是在物理层面上，分出来的一个操作系统；

多个容器可以运行在同一台物理服务器上，并共享一个操作系统的内核资源。多个虚拟机也可以运行在同一台机器上，但每个虚拟机都需

要一个完整的操作系统；

下图比较了 Docker 和传统虚拟化方式的不同之处：





可见容器是在本地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，不需要单独安装操作系统，而传统的虚拟化技术方式则需要单独安装每个虚拟机的操作系统。

特性	**容器**	**虚拟机**
启动	秒级	分钟级
硬盘空间	一般为几十 MB	一般为 10GB
性能	接近原生	弱于原生
系统支持量	单机支持上千个容器	一般几十个

操作系统	与宿主机共享 OS	宿主机 OS 上运行虚拟机 OS

1-9 为什么使用 Docker

作为一种新兴的虚拟化技术，Docker 跟传统的虚拟化方式相比具有众多的优势。

1、Docker 容器的启动可以在秒级实现，这相比传统的虚拟机方式要快得多。

2、Docker 对系统资源的利用率很高，一台主机上可以同时运行数千个 Docker 容器。

3、容器除了运行其中的应用外，基本不消耗额外的系统资源，使得应用的性能很高。传统虚拟机方式运行 10 个完全不同的应用可能我们会起 10 个虚拟机来部署，而 Docker 只需要启动 10 个隔离的应用即可。

4、Docker 可以更快速的交付和部署，大量地节约开发、测试、部署的时间，对开发和运维人员来说，最希望的就是一次创建或配置，可以在任意地方正常运行。

5、更高效的虚拟化，Docker 容器的运行不需要额外的 hypervisor 支持，它是内核级的虚拟化，因此可以实现更高的性能和效率。

6、更轻松的迁移和扩展，Docker 容器几乎可以在任意的平台上运行，包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等，这种

兼容性可以让用户轻松地把一个应用程序从一个平台直接迁移到另一个平台。

第二章 Docker 环境搭建

2-1 Docker 的版本

Docker 从 2013 年 3 月 20 日发布 Docker 0.1，到现在已经发布了多个版本，从 2017 年 3 月开始 docker 在原来的基础上分为两个分支版本: Docker CE 和 Docker EE。

Docker CE 即社区免费版，可永久免费使用；

Docker EE 即企业版，功能更全，更强调安全，但需付费使用；

本课程介绍 Docker CE 版本，目前 Docker 版本为 18.03

Docker 官方网站：https://www.docker.com/

2-2 Docker 的安装

首先，我们知道 Docker 并不是容器，它是一个管理容器的引擎。

我们课程采用的 Linux 版本是 CentOS 7,学习 Docker 也更推荐在 Linux 环境下使用；

Docker 支持 CentOS 6 及以后的版本；

CentOS7 系统可以直接通过 yum 进行安装：

安装前可以查看一下系统是否已经安装了 Docker：

yum list installed | grep docker

安装：yum install docker -y

安装后，使用 docker –version（docker version，docker -v）查看 docker 是否安装成功，

卸载：

yum remove docker.x86_64 -y

yum remove docker-client.x86_64 -y

yum remove docker-common.x86_64 -y

2-3 Docker 服务启动

安装之后启动 Docker 服务；

启动：systemctl start docker 或者 service docker start

停止：systemctl stop docker 或者 service docker stop

重启：systemctl restart docker 或者 service docker restart

检查 docker 进程的运行状态：

systemctl status docker 或者 service docker status

查看 docker 进程：ps -ef | grep docker

2-4 Docker 服务信息

docker info 查看 docker 系统信息

docker 查看所有的帮助信息

`docker commond --help` 查看某个 commond 命令的帮助信息

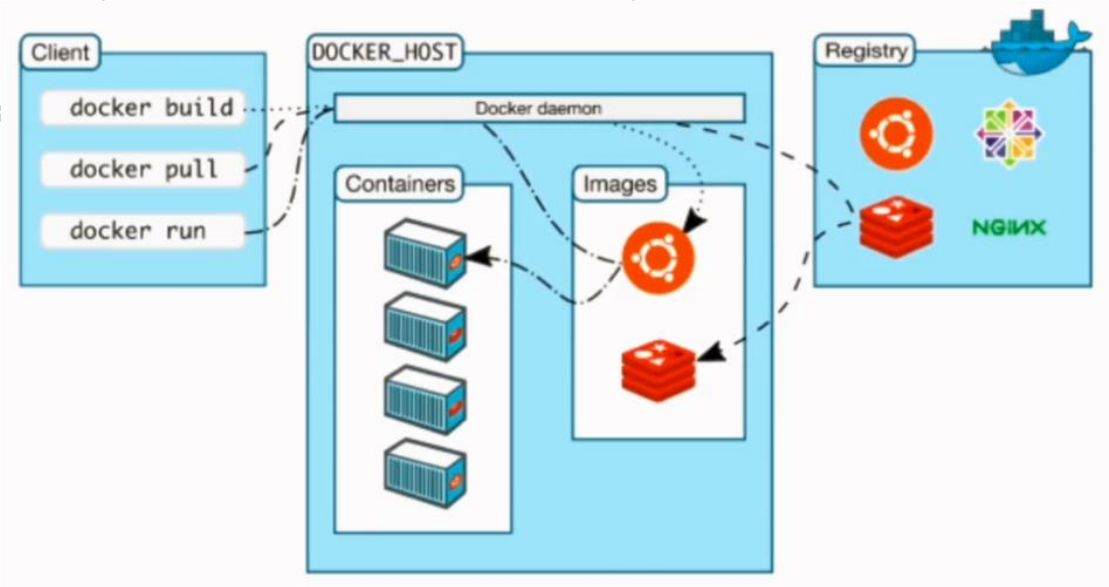
2-5 Docker 使用初体验

2-5-1 Docker 的运行机制

我们知道 Docker 并不是容器，而只是一个管理容器的引擎；

Docker 的底层运行原理：

Docker 服务启动→下载镜像→启动该镜像得到一个容器→容器里运行着我们想要的程序；



2-5-2 第一个 Docker 容器

根据 Docker 的运行机制，我们将按照如下步骤运行第一个 Docker 容器；

1、将 Docker 服务启动；

2、下载一个镜像，Docker 运行一个容器前需要本地存在有对应的镜像，如果镜像不存在本地，Docker 会从镜像仓库下载（默认是 Docker Hub 公共注册服务器中的仓库 https://hub.docker.com）。

CentOS 下怎么下载（pull）镜像?

从 docker hub 官网搜索要使用的镜像，也可以在命令行使用命令搜索要使用的镜像，比如 docker search tomcat 进行搜索，然后下载所需要的镜像：

下载镜像：docker pull tomcat

运行镜像：docker run tomcat 前台运行， 要后台运行，加参数 -d

显示本地已有的镜像：docker images

```
[root@bogon /]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker.io/tomcat    latest              41a54fe1f79d        4 weeks ago         463 MB
```

在列出信息中，可以看到几个字段信息

REPOSITORY：来自于哪个仓库，比如 docker.io/tomcat

TAG：镜像的标记，比如 latest

IMAGE ID：镜像的 ID 号（唯一）

CREATED：创建时间

SIZE：镜像大小

3、启动下载下来的镜像得到一个容器：

docker run -d docker.io/tomcat 或者 docker run -d 41a54fe1f79d

默认是前台启动，如果需要后台启动，指定-d 参数；

通过 ps -ef | grep tomcat 查看，检查 tomcat 镜像是否启动容器成功；

2-5-3 进入 Docker 容器

进入容器：docker exec -it cef0d139bfd6 bash

其中 i 表示交互式的，也就是保持标准输入流打开；

t 表示虚拟控制台，分配到一个虚拟控制台；

退出容器：exit

2-5-4 客户机访问容器

从客户机上访问容器，需要有端口映射，docker 容器默认采用桥接模式与宿主机通信，需要将宿主机的 ip 端口映射到容器的 ip 端口上；

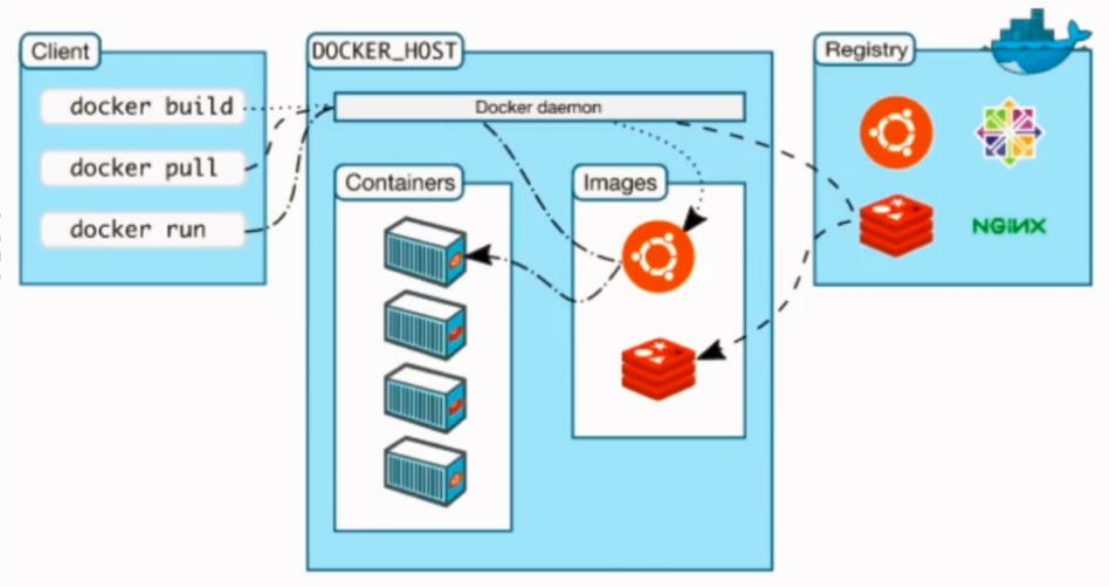
停止容器：docker stop 容器 ID/名称

启动容器：docker run -d -p 8080:8080 docker.io/tomcat 或者 41a54fe1f79d

第三章 Docker 核心组件

3-1 Docker 架构

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程 API 来管理和创建 Docker 容器。



Docker 容器通过 Docker 镜像来创建。

镜像与容器的关系类似于面向对象编程中的类与对象的关系。

Docker	面向对象
镜像	类
容器	对象

3-2 Docker 核心要素

Docker 包括三个核心要素

镜像（Image）、容器（Container）、仓库（Repository）

理解了这三个概念，就理解了 Docker 的整个生命周期。

Docker 的运行离不开以上核心几个组件的支持，Docker 的成功也是拜这几个组件所赐。

有人会误以为，Docker 就是容器，但 Docker 不是容器，而是管理容器的引擎。

3-3 镜像

3-3-1 镜像的基本概念

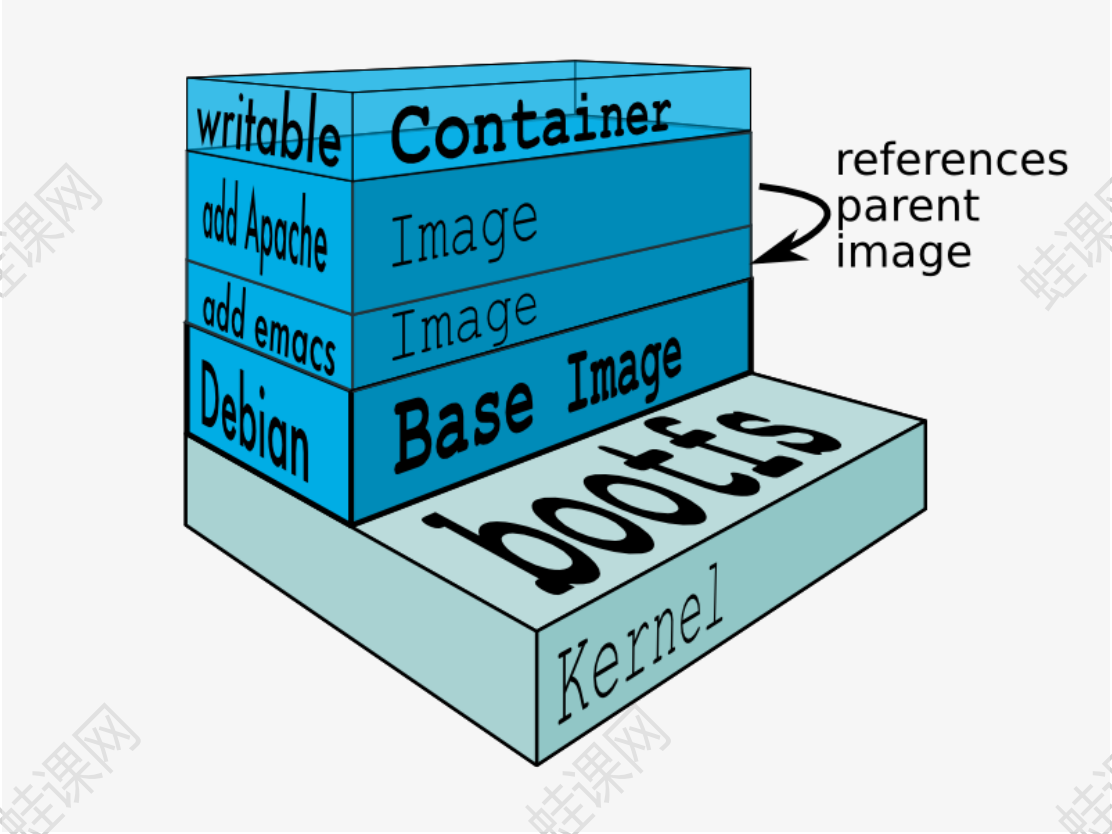
Docker 镜像就是一个只读的模板，可以用来创建 Docker 容器。

例如：一个镜像可以包含一个完整的 centos 操作系统环境，里面仅安装了 mysql 或用户需要的其它应用程序。

Docker 提供了一个非常简单的机制来创建镜像或者更新现有的镜像，用户甚至可以直接从其他人那里下载一个已经做好的镜像来直接使用。

3-3-2 镜像的组成结构

镜像是由许多层的文件系统叠加构成的，最下面是一个引导文件系统 bootfs，第二层是一个 root 文件系统 rootfs，root 文件系统通常是某种操作系统，比如 centos、Ubuntu，在 root 文件系统之上又有很多层文件系统，这些文件系统叠加在一起，构成 docker 中的镜像；



3-3-3 镜像的日常操作

1、下载镜像，比如下载 redis 镜像：docker pull redis:latest

reids 是查询到的镜像名称，latest 是镜像的标签 tag

获取一个镜像有两种方式，一种是从官方镜像仓库下载，一种是自己通过 Dockerfile 文件构建。

如果有官方镜像，我们就不必自己用 Dockerfile 文件构建了，除非官方没有才会自己去 Dockerfile 文件构建；

2、列出已经下载的镜像：docker images，或者 docker images redis

3、运行镜像：docker run -d redis 其中-d 表示在后台运行

然后通过 ps -ef | grep redis 可以查到 redis 进程

4、查看容器镜像的状态：docker ps

通过 docker exec -it a8584016f9b6(镜像 ID) bash 进入 redis 容器

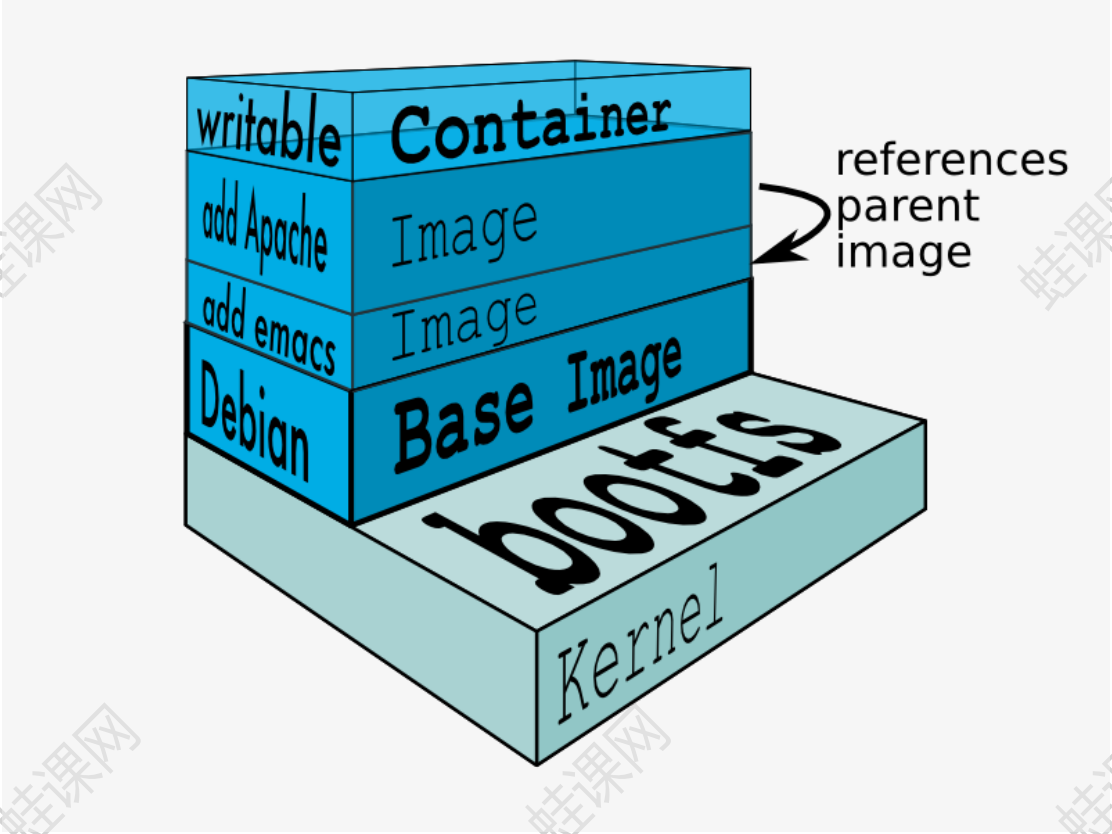
5、删除镜像：docker rmi redis:latest 注意是 rmi，不是 rm，rm 是删除容器；

3-4 容器

3-4-1 容器的基本概念

容器是从镜像创建的运行实例。它可以被启动、停止、删除。每个容器都是相互隔离的、保证安全平台。可以把看做一个简易版的 Linux 环境，包括 root 用户权限、进程空间、用户空间和网络空间和运行在其中的应用程序。

Docker 利用容器来运行应用，镜像是只读的，容器在启动的时候创建一层可写层作为最上层。



3-4-2 容器的日常操作

启动容器有两种方式，一种是基于镜像新建一个容器并启动，另外一个是将在终止状态的容器重新启动。

通过镜像启动容器：docker run -d redis

查看运行中的容器：docker ps

查看所有的容器：docker ps -a

停止容器：docker stop 容器 id 或容器名称

已经停止的容器，我们可以使用命令 docker start 来启动。

开启容器：docker start 容器 id 或容器名称

因为 Docker 的容器实在太轻量级了，很多时候用户都是随时删除和

新创建容器。

删除容器：docker rm 容器 id 或容器名称

删除容器时，容器必须是停止状态，否则会报错；

进入容器：docker exec -it 容器 id 或容器名称 bash

还可以使用 docker inspect + 容器 id 或容器名称 查看容器的更多信息；

停用全部运行中的容器：docker stop $(docker ps -q)

删除全部容器：docker rm $(docker ps -aq)

一条命令实现停用并删除容器：

docker stop $(docker ps -q) & docker rm -f $(docker ps -aq)

3-5 仓库

3-5-1 仓库的基本概念

仓库是集中存放镜像文件的场所，有时候会把仓库和仓库注册服务器（Registry）看做同一事物，并不严格区分。实际上，仓库注册服务器上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（tag）；

仓库分为公开仓库（Public）和私有仓库（Private）两种形式；

最大的公开仓库是 Docker Hub （https://hub.docker.com/），存放了数量庞大的镜像供用户下载；

当然，用户也可以在本地网络内创建一个私有仓库；

当用户创建了自己的镜像之后就可以使用 push 命令将它上传到公有或者私有仓库，这样下次在另外一台机器上使用这个镜像时候，只需要从仓库上 pull 下来即可；

注：Docker 仓库的概念跟 Git 类似，注册服务器也类似于 GitHub 这样的托管服务；

3-5-2 仓库的日常操作

用户可通过 docker search 命令来查找官方仓库中的镜像：

```
docker search rabbitmq
```

可以看到返回了很多包含关键字的镜像，其中包括镜像名字、描述、星级（表示该镜像的受欢迎程度）、是否官方创建、是否自动创建；

官方的镜像说明是官方项目组创建和维护的，automated 资源允许用户验证镜像的来源和内容；

根据是否是官方提供，可将镜像资源分为两类；

一种是类似 centos 这样的基础镜像，被称为基础或根镜像。这些基础镜像是由 Docker 公司创建、验证、支持、提供。这样的镜像往往使用单个单词作为名字；

还有一种类型，比如 tianon/centos 镜像，它是由 Docker 的用户创建并维护的，往往带有用户名称前缀。可以通过前缀 user_name/ 来指定使用某个用户提供的镜像，比如 tianon 用户；

并利用 docker pull 命令来将它下载到本地：

docker pull rabbitmq

docker pull centos

第四章 Docker 使用示例

4-1 Docker 安装 MySQL

#下载 MySQL 镜像：

docker pull mysql:latest （安装的是 mysql 8.0）

docker run -p 3306:3306 -e MYSQL_DATABASE=workdb -e MYSQL_ROOT_PASSWORD=123456 -d mysql:latest

其中-e 是指定环境变量

#进入容器：

docker exec -it 3e8bf7392b4e bash

#登录 MySQL：

mysql -u root -p

#修改密码：

ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';

#授权远程登录访问：

CREATE USER 'wkcto'@'%' IDENTIFIED WITH mysql_native_password BY '123456';

GRANT ALL PRIVILEGES ON *.* TO ' wkcto'@'%';

4-2 Docker 安装 Nginx

#下载 Nginx 镜像：

docker pull nginx

docker run -d -p 80:80 nginx

#进入容器：

docker exec -it 3e8bf7392b4e bash

#浏览器访问 Nginx：

http://192.168.230.128:80

#Nginx 部署静态网站：

将 linux 的文件拷贝到 docker 容器某个目录下：

docker cp /root/test.html bf8a58328e18:/usr/share/nginx/html

4-3 Dokcer 安装 Zookeeper

#下载 Zookeeper 镜像：

docker pull zookeeper

docker run -p 2181:2181 -d zookeeper

#进入容器：

docker exec -it 3e8bf7392b4e bash

#客户端工具访问 Zookeeper：

4-4 Docker 安装 ActiveMQ

#下载 ActiveMQ 镜像：

docker pull webcenter/activemq

docker run -p 8161:8161 -d activemq

#进入容器：

docker exec -it 3e8bf7392b4e bash

#浏览器访问 activemq：

第五章 Docker 自定义镜像

5-1 认识 Dockerfile 文件

Dockerfile 用于构建 Docker 镜像，Dockerfile 文件是由一行行命令语句组成，基于这些命令即可以构建一个镜像，比如下面就是一个 Dockefile 文件样例：

```
FROM XXX/jdk:8
MAINTAINER docker_user
ENV JAVA_HOME /usr/local/java
ADD apache-tomcat-8.0.32.tar.gz /usr/local/
RUN mv apache-tomcat-8.0.32 tomcat8
EXPOSE 8080
RUN chmod u+x /usr/local/tomcat8/bin/*.sh
CMD /usr/local/tomcat8/bin/catalina.sh start
```

5-2 Dockerfile 的基本结构

一般的，Dockerfile 分为四部分：

基础镜像信息；

维护者信息；

镜像操作指令；

容器启动时执行指令；

5-3 Dockerfile 指令

FROM

格式为 FROM <image> 或 FROM <image>:<tag>

Dockerfile 文件的第一条指令必须为 FROM 指令。并且，如果在同一个 Dockerfile 中创建多个镜像时，可以使用多个 FROM 指令（每个镜像一次）；

MAINTAINER

格式为 MAINTAINER <name>，指定维护者信息；

ENV

格式为 ENV <key> <value>，指定一个环境变量，会被后续 RUN 指令使用，并在容器运行时保持；

ADD

格式为 ADD <src> <dest>；

复制指定的<src>到容器中的<dest>；

EXPOSE

格式为 EXPOSE <port> [<port>...]

告诉 Docker 服务端容器暴露的端口号，供互联系统使用，在启动容器时需要通过 -p 映射端口，Docker 主机会自动分配一个端口转发到指定的端口；

RUN

格式为 RUN <command>

RUN 指令将在当前镜像基础上执行指定命令，并提交为新的镜像，当命令较长时可以使用 \ 来换行；

CMD

指定启动容器时执行的命令，每个 Dockerfile 只能有一条 CMD 命令。如果指定了多条命令，只有最后一条会被执行。

如果用户启动容器时候指定了运行的命令，则会覆盖掉 CMD 指定的命令。

5-4 Dockerfile 自定义镜像

5-4-1 自定义 JDK 镜像

```
FROM centos:latest
MAINTAINER wkcto
ADD jdk-8u121-linux-x64.tar.gz /usr/local
ENV JAVA_HOME /usr/local/jdk1.8.0_121
ENV CLASSPATH $JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
ENV PATH $PATH:$JAVA_HOME/bin
CMD java -version
```

构建镜像：docker build -t wkcto_jdk1.8.0_121 .

运行镜像：docker run -d ac84bde53958

5-4-2 自定义 Tomcat 镜像

定义 Dockerfile 文件：

```
FROM wkcto_jdk1.8.0_121
MAINTAINER wkcto
ADD apache-tomcat-8.5.24.tar.gz /usr/local/
ENV CATALINA_HOME /usr/local/apache-tomcat-8.5.24
ENV PATH $PATH:$CATALINA_HOME/lib:$CATALINA_HOME/bin
EXPOSE 8080
CMD /usr/local/apache-tomcat-8.5.24/bin/catalina.sh run
```

构建镜像：docker build -t wkcto-tomcat-8.5.24 .

运行镜像：docker run -d -p 8080:8080 ab41b5f48256

5-4-3 自定义 MySQL 镜像

定义 Dockerfile 文件：

```
FROM centos:centos6
MAINTAINER wkcto
RUN yum install mysql-server mysql -y
RUN /etc/init.d/mysqld start &&\
    mysql -e "grant all privileges on *.* to 'root'@'%' identified by '123456' WITH GRANT OPTION ;"&&\
    mysql -e "grant all privileges on *.* to 'root'@'localhost' identified by '123456' WITH GRANT OPTION ;"&&\
    mysql -uroot -p123456 -e "show databases;"
EXPOSE 3306
CMD /usr/bin/mysqld_safe
```

构建镜像：docker build -t wkcto-mysql .

运行镜像：docker run -d -p 3306:3306 09ce279d92df

5-4-4 自定义 Redis 镜像

定义 Dockerfile 文件：

```
FROM centos:latest
MAINTAINER wkcto
RUN yum install epel-release -y && yum install redis -y && yum install net-tools -y
EXPOSE 6379
CMD /usr/bin/redis-server –protected-mode no
```

构建镜像：`docker build -t wkcto-redis .`

运行镜像：`docker run -d -p 6379:6379 390583cf0531`

5-5 镜像发布到仓库

5-5-1 阿里云容器镜像仓库

网址：https://dev.aliyun.com

5-5-2 注册阿里云镜像仓库

如果没有阿里云账号，需要注册一个阿里云账号；

也可以使用淘宝账号、支付宝账号登录；

5-5-3 发布镜像阿里云镜像仓库

登录阿里云镜像，进入管理中心，在自己我管理中心中，左侧点击镜像仓库菜单，如果自己没有镜像仓库的话，需要创建一个镜像仓库；对自己的镜像仓库点击管理链接，按照操作指南进行操作即可；

1. 登录阿里云 Docker Registry

```
docker login --username=hi35331710@aliyun.com registry.cn-qingdao.aliyuncs.com
```

用于登录的用户名为阿里云账号全名，密码为开通服务时设置的密码。

您可以在产品控制台首页修改登录密码。

```
[root@localhost docker]# docker login --username=hi35331710@aliyun.com registry.cn-qingdao.aliyuncs.com
Password:
Login Succeeded
```

2. 将镜像推送到 Registry

docker tag [ImageId] registry.cn-qingdao.aliyuncs.com/123test/1234test:[镜像版本号]

docker push registry.cn-qingdao.aliyuncs.com/123test/1234test:[镜像版本号]

请根据实际镜像信息替换示例中的[ImageId]和[镜像版本号]参数。

imageId: a78da0202f84

docker tag a78da0202f84 registry.cn-qingdao.aliyuncs.com/123test/1234test:latest

docker push registry.cn-qingdao.aliyuncs.com/123test/1234test:latest

```
[root@localhost docker]#
[root@localhost docker]# docker tag baee2122a5f6 registry.cn-qingdao.aliyuncs.com/123test/1234test:latest
[root@localhost docker]#
[root@localhost docker]# docker push registry.cn-qingdao.aliyuncs.com/123test/1234test:latest
The push refers to a repository [registry.cn-qingdao.aliyuncs.com/123test/1234test]
796d7b1d79bf: Pushed
698ac9e7c1eb: Pushed
dc8353e32a29: Pushed
22df72639f8c: Pushed
8149340ee0ca: Pushed
7be5ad076570: Pushed
81d790f1eee8: Pushed
27204ff62926: Pushed
c65085b40b81: Pushed
ee85b588d1bf: Pushed
12a964b08439: Pushed
38315d1ca903: Pushed
8b15606a9e3e: Pushed
latest: digest: sha256:ef3f87d6c203f7bb7acf0768001a66aeda1e12bdd3f072c63b0df09240cf2774 size: 3035
[root@localhost docker]#
```

5-5-4 Docker hub 镜像加速

/etc/docker/daemon.json

{"registry-mirrors": ["https://gg3gwnry.mirror.aliyuncs.com"]}

第六章 Docker 应用部署

6-1 部署一个 SpringBoot 项目

1、将开发的 springboot 程序打成 jar 包或者 war 包；

2、将打好的 jar 包或 war 包上传到 Linux 某个目录下，比如:/root/docker

3、定义 Dockerfile 文件，用于创建项目镜像；

6-2 Docker 部署 Jar 包 SpringBoot 程序

6-2-1 定义 Jar 包程序 Dockerfile 文件

```
FROM wkcto_jdk1.8.0_121
MAINTAINER wkcto
ADD springboot-web-1.0.0.jar /opt
RUN chmod +x /opt/springboot-web-1.0.0.jar
CMD java -jar /opt/springboot-web-1.0.0.jar
```

6-2-2 构建和运行 Jar 包程序的镜像

构建镜像：docker build -t springboot-web-jar .

运行容器：docker run -d ac84bde53958

6-2-3 Jar 包程序依赖容器环境准备

运行 Redis 容器：docker run -p 6379:6379 -d redis

运行 MySQL 容器：docker run -p 3306:3306 -e MYSQL_DATABASE=workdb -e MYSQL_ROOT_PASSWORD=123456 -d mysql:latest

修改容器保存：docker commit 容器 id xxx(镜像名:tagxxx)

比如：docker commit b034f6d23833 wkcto_mysql_new

6-2-4 运行 Docker 化的 Jar 包程序

通过 windows 的浏览器访问，验证 SpringBoot 项目是否可以正常访问；

6-3 Docker 部署 War 包 SpringBoot 程序

6-3-1 定义 War 包程序 Dockerfile 文件

```
FROM wkcto-tomcat-8.5.24
MAINTAINER wkcto
ADD springboot-web-1.0.0.war /usr/local/apache-tomcat-8.5.24/webapps
EXPOSE 8080
CMD /usr/local/apache-tomcat-8.5.24/bin/catalina.sh run
```

6-3-2 构建和运行 War 包程序的镜像

构建镜像：docker build -t springboot-web-war .

6-3-3 War 包程序依赖容器环境准备

与上面部署 Jar 程序依赖的容器环境一样；

6-3-4 运行 Docker 化的 War 包程序

通过 windows 的浏览器访问，验证 SpringBoot 项目是否可以正常访问；

springboot-web-1.0.0.war → springboot-web-1.0.0