容器技术概述

前言

- 本章开篇介绍容器技术的发展历史，以便使学员了解容器技术发展的背景及其解决的问题。而后介绍容器技术的核心概念并演示容器的基本操作维护。

目标

- 学完本课程后，您将能够：
  - 描述容器技术的发展
  - 描述容器技术的核心概念和优势
  - 掌握容器技术的基础操作、生命周期管理

目录

应用上云的痛点

- 云计算重构了ICT系统，给社会各行业带来了极大的变革、便利。

- 但云计算同样带来了新的问题：业务怎么上云？
  - 应用在云端重新部署：以脚本或手工方式在云端重新部署。
  - 打包本地已部署应用的系统镜像，通过P2V/V2V等方式上传到云端运行。

Cloud Foundry项目介绍

- Cloud Foundry等传统开源PaaS项目的作用
  - 用于解决大规模的应用“上云“的问题。Cloud Foundry项目提供了**“应用托管”**能力，其核心组件是一套应用的打包和分发机制。
- Cloud Foundry项目的实现过程：
  - **自动上传、运行应用包**：运维人员需在本地物理机或虚拟机上部署一个Cloud Foundry项目。开发人员执行简单命令后，即可将本地应用（应用的可执行文件和启动脚本）打包进一个压缩包内，上传到Cloud Foundry云端存储着。Cloud Foundry会通过调度器选择一个可运行该应用的虚拟机并通知其agent下载该应用的压缩包，然后启动运行应用。
  - **为应用提供隔离的运行环境**：由于一个虚拟机中可能需运行来自多个用户的不同应用，Cloud Foundry会调用Namespace和Cgroup为每个应用创建隔离的运行环境（沙盒），以实现应用间互不干涉。在这点上，与Docker Container的实现类似。

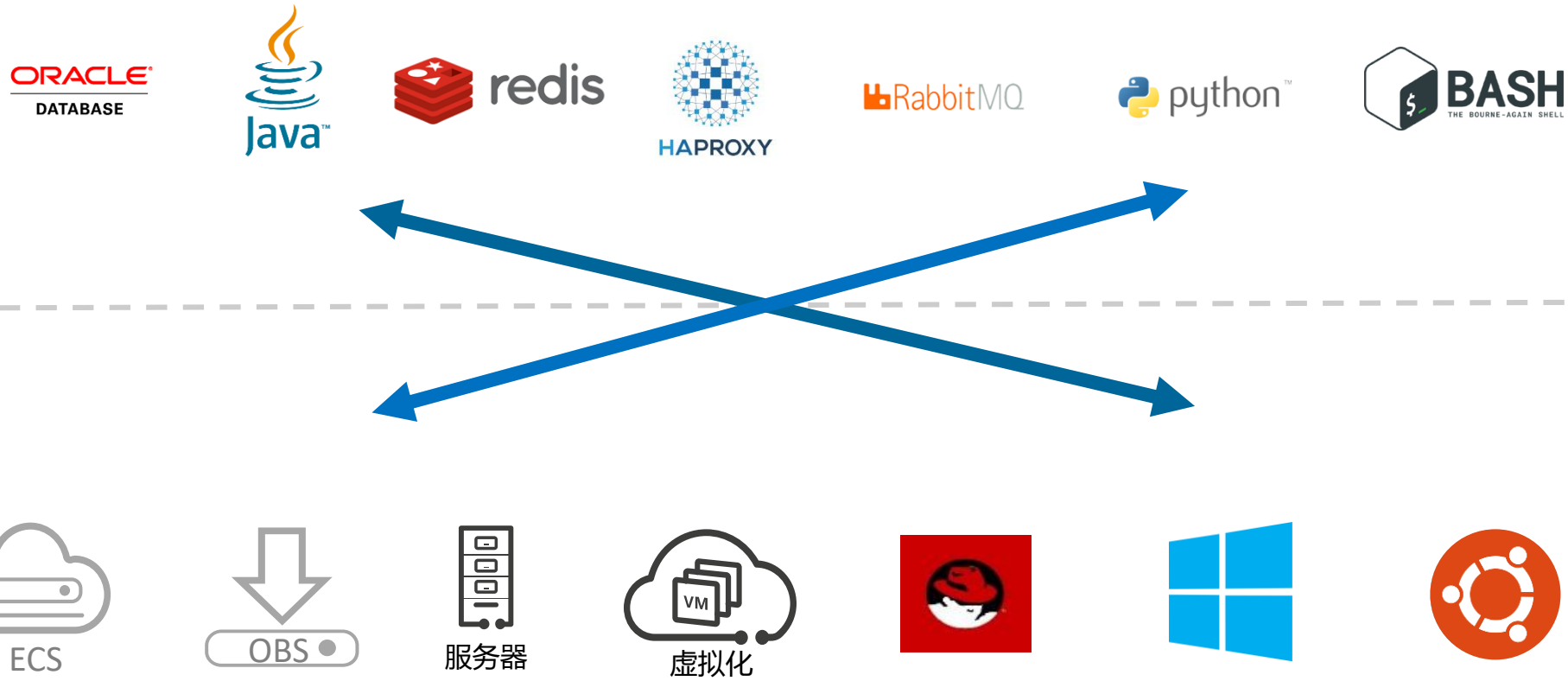
SandBox

传统PaaS项目的痛点

- 传统开源PaaS项目中的问题：
  - 应用打包困难：因本地环境与云端环境不一致，用户须为每种语言、框架乃至每个版本的应用维护一个打好的包。而打包过程中，需要进行大量修改、配置、试错才能使本地应用运行环境和云端环境匹配。

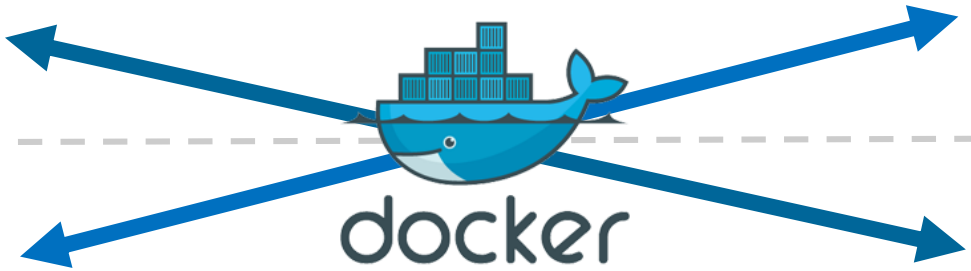
一键部署上云运行很痛快，打包过程很痛苦。

传统IT应用的挑战



Docker项目的创新

- Docker镜像：
  - 容器镜像打包了应用及其依赖（包含完整操作系统的所有文件和目录）。
  - 容器镜像包含了应用运行所需要的所有依赖。只需在隔离的“沙盒”中运行该镜像，无需进行任何修改和配置即可运行应用。
  - 容 在于实现应用及其运行环境整体 式统一。实现本地环境与云端环境的高一致性。

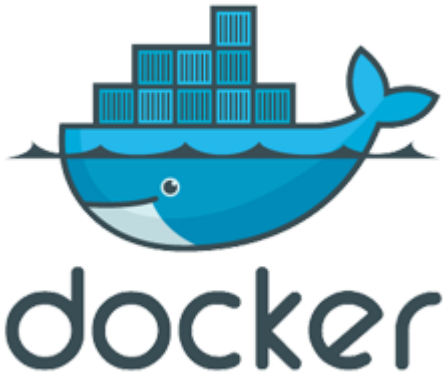


Docker项目介绍

- 2013年，dotCloud公司将Docker项目开源。
- Docker项目：
  - GitHub上开发的Moby开源项目的一部分。
  - 遵循Apache License 2.0许可证协议。
  - Go语言编写。

- Docker是一个开源的引擎，可以轻松的为任何应用创建一个轻量级的、可移植的、自给自足的容器。

- Docker公司目前推出两个版本：
  - Docker CE（社区版）
  - Docker EE（企业版）



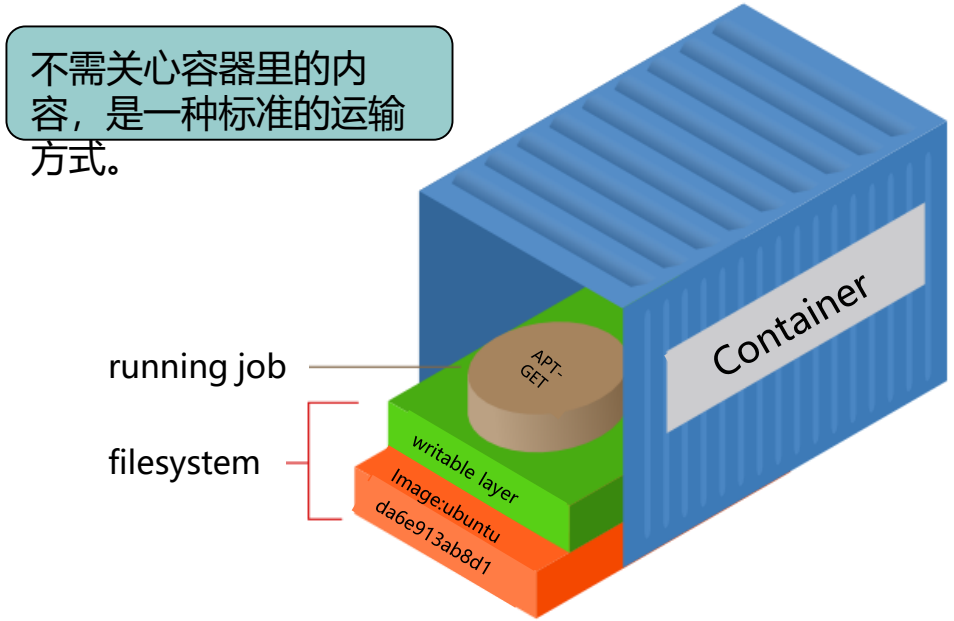
旧logo

新logo

容器是什么

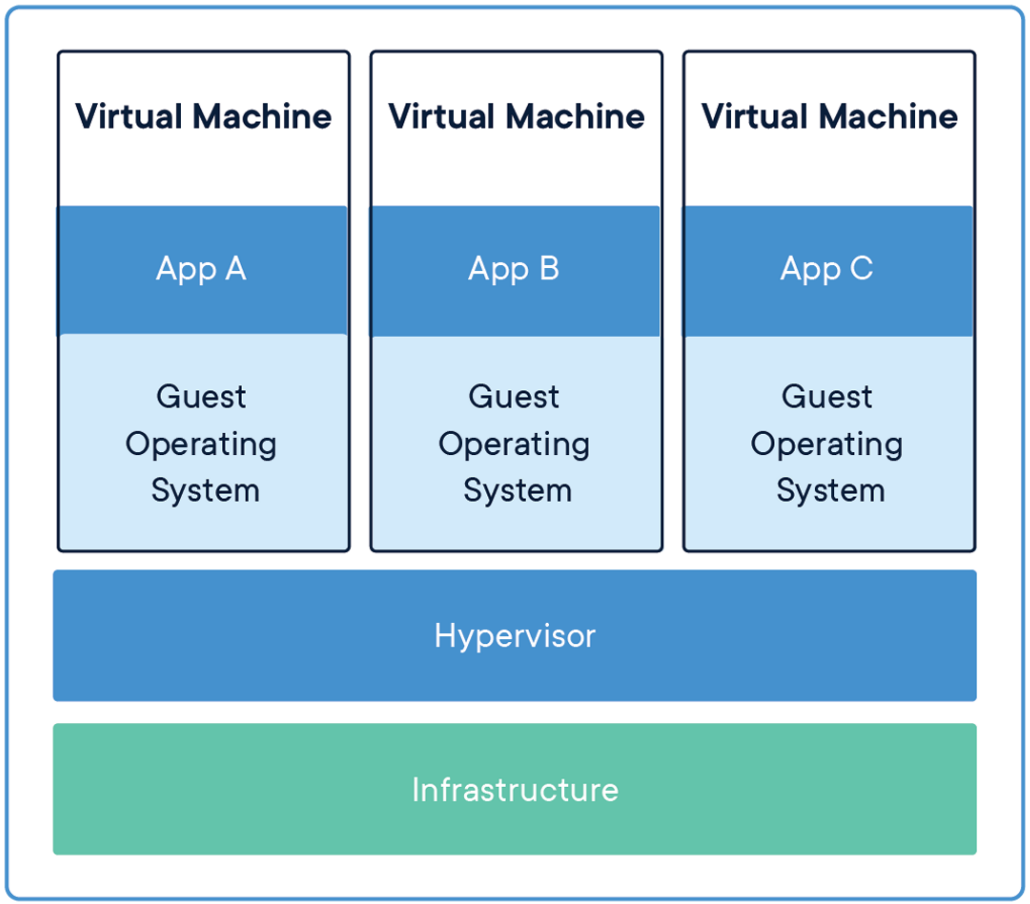
- 定义：容器是容器image运行时的实例。
- 通俗的理解：软件界的集装箱（隔离、封装）。

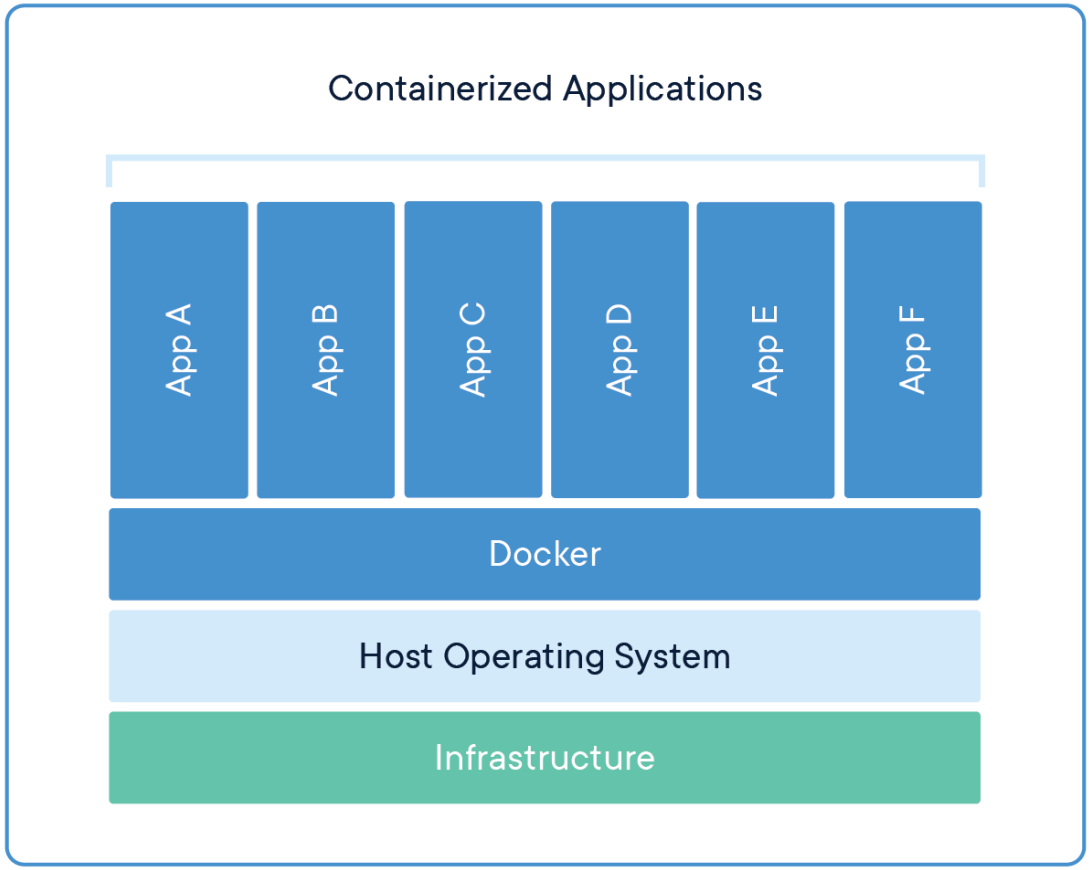
运输业



Container

Container VS VM (1)





Container VS VM (2)

	容器	虚拟机
启动速度	秒甚至毫秒启动	数秒至数十秒
系统内核	共享内核	不共享内核
实现技术	利用Linux内核技术Namespace/Cgroup等实现。	依赖虚拟化技术实现，由Hypervisor层实现对资源的隔离
隔离效果	进程级别的隔离	系统资源级别的隔离
资源消耗（性能）	容器中的应用只是宿主机上的一个普通进程	使用虚拟化技术，就会有额外的资源消耗和占用
资源调用（敏捷性）	应用进程直接由宿主机OS管理	应用进程需经过Hypervisor的拦截和处理，才能调用系统资源
运行数量	一台服务器上能启动1000+容器	一台服务器上一般不超过100台虚拟机
应用	DevOps、微服务等	用于硬件资源划分
镜像	分层镜像	非分层镜像

容器编排引擎

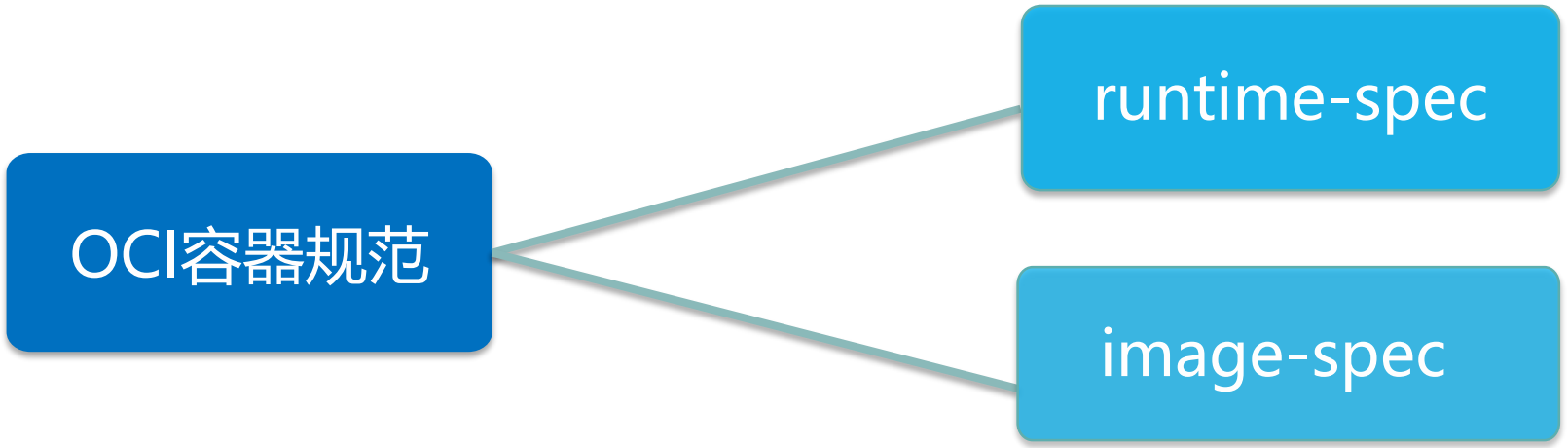
CNCF基金会

- CNCF基金会：Cloud Native Computing Foundation
  - 隶属于Linux Foundation
  - 致力于云原生（Cloud Native）技术的普及和可持续发展。
- CNCF基金会下的明星项目
  - Kubernetes
  - Prometheus
  - Fluentd
  - CNI
  - gPRC

目录

OCI容器规范

- OCI：Open Container Initiative
  - 隶属于Linux Foundation
  - 旨在创建容器格式和容器runtime的开放行业标准



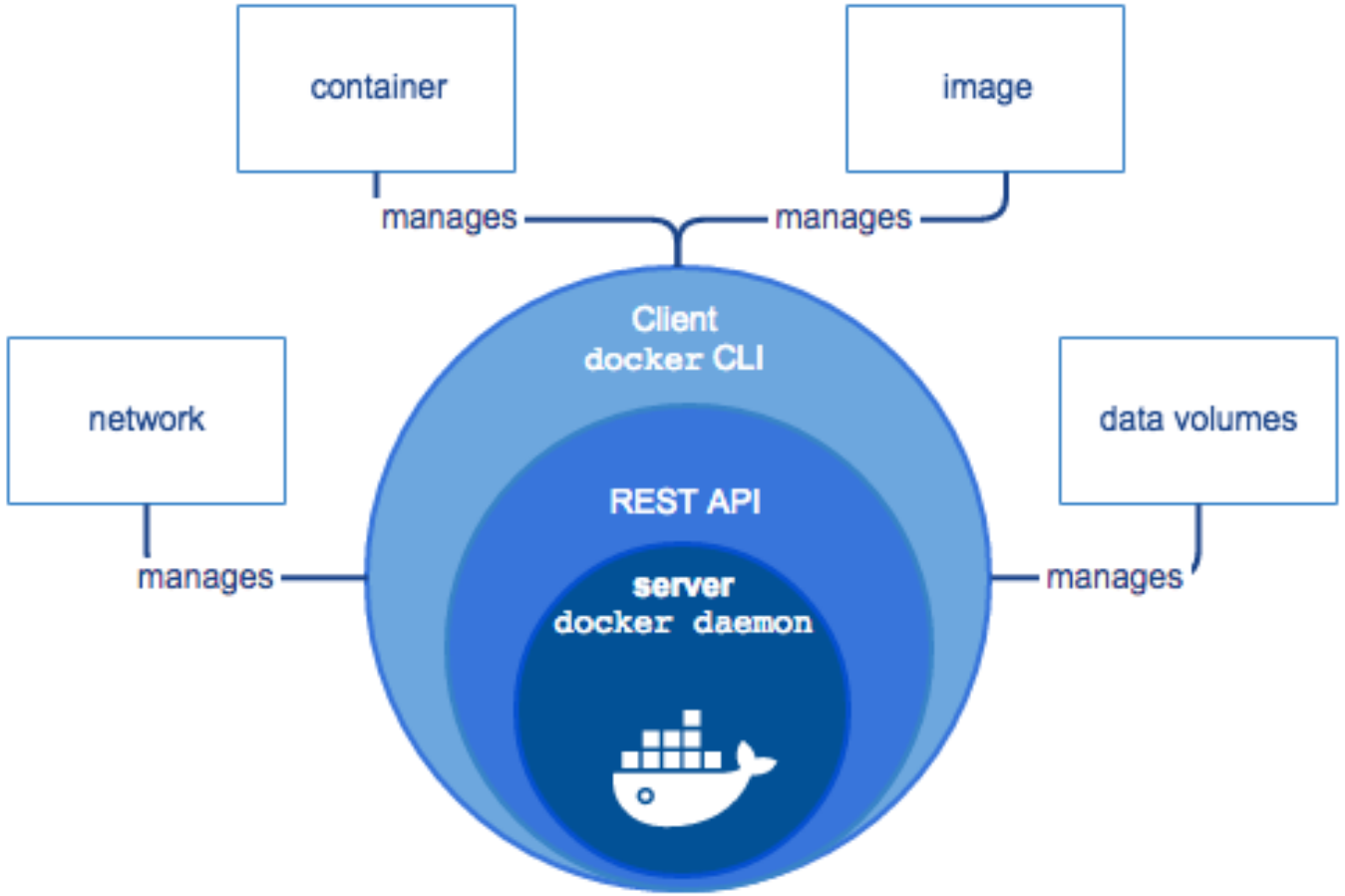
Runtime

- OCI下的Runtime规范旨在指定容器的配置、执行环境和生命周期。
- Runtime是对容器运行时的相关状态和操作进行管理的工具。Runtime定义了如下规范对容器进行管理：

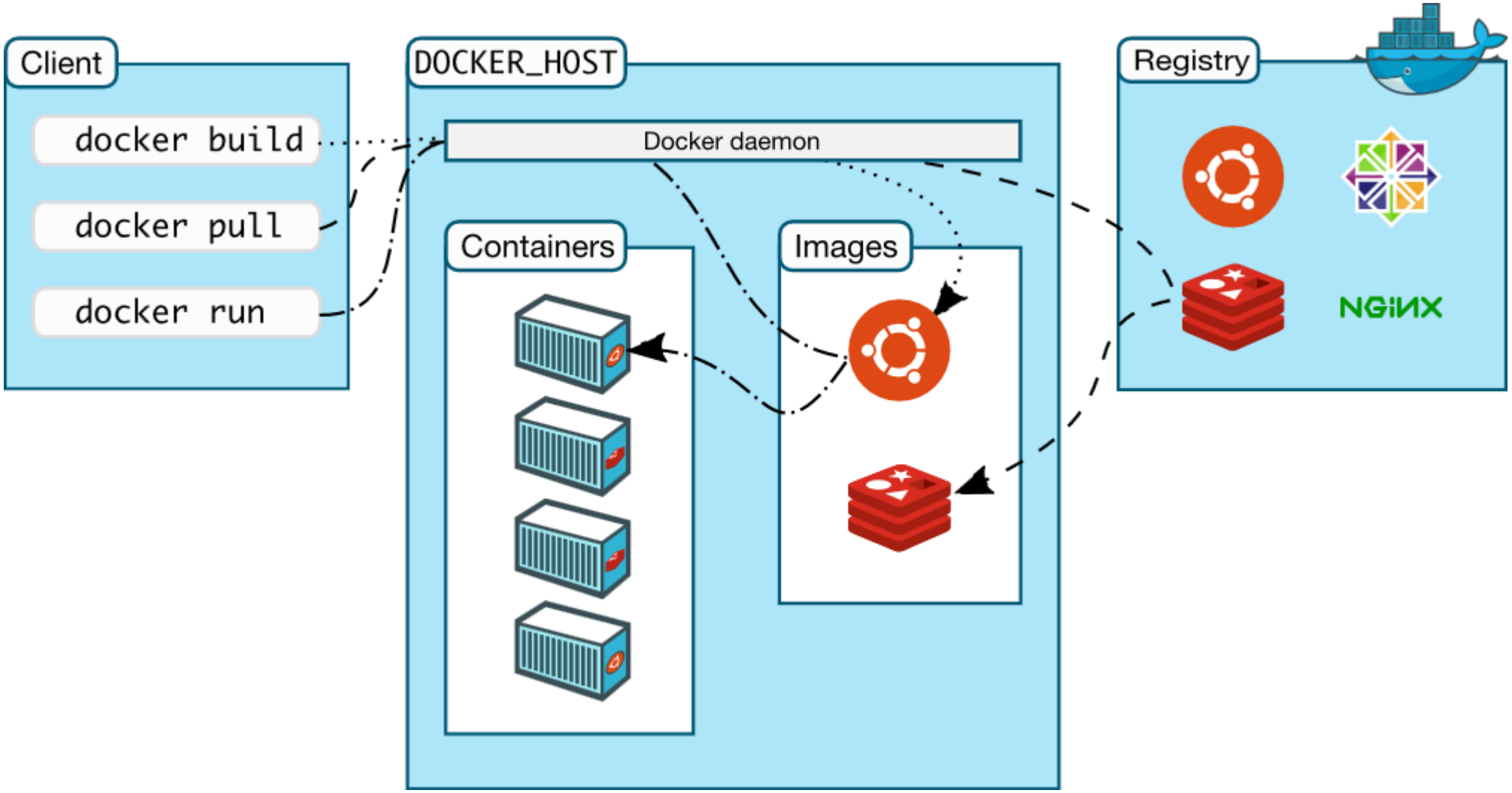
	内容
bundle.md (Filesystem Bundle)	bundle中包含了运行容器所需要的所有信息，主要是config.json文件和rootfs。Runtime根据bundle启动容器。
config.md	包含对容器实施标准操作所必需的元数据，存放于config.json文件中。如oci版本、rootfs路径、mount目录、process、platform、容器hostname等。
config-linux.md	Linux平台上对config.md的扩展，内容也包含在config.json文件中。如namespace、devices、cgroupPath、resources、sysctl、readonlyPaths等。
runtime.md	定义了3部分内容：容器状态（如status、pid等）、容器相关操作（如create、kill等）、容器生命周期。
runtime-linux.md	是Linux平台上对runtime.md的补充。

- Runtime工具
  - runC是Docker公司2015年发布的符合OCI规范的runtime工具。
    - runC由Libcontainer演变而来。
  - 业界其他runtime工具有Linux上的LXC，CoreOS的rkt等。

Docker Engine介绍



Docker架构



查看docker服务状态

- 查看Docker engine状态

```
[root@localhost /]# systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2019-07-16 08:52:39 EDT; 1 weeks 3 days ago
     Docs: https://docs.docker.com
 Main PID: 73043 (dockerd)
    Tasks: 35
   Memory: 299.6M
   CGroup: /system.slice/docker.service
           ├─73043 /usr/bin/dockerd
           ├─73049 docker-containerd --config /var/run/docker/containerd/containerd.toml
           ├─88328 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8080 -container-ip 172.17.0.2 -container-port 80
           └─88333 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1.lir
```

目录

运行一个容器 (1)

- 执行docker run命令运行一个容器。
  - “-d”参数可在后台运行容器；“-p”参数将宿主机8080端口映射到容器80端口。

```
[root@localhost ~]# docker run -d -p 8080:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
f5d23c7fed46: Pull complete
b083c5fd185b: Pull complete
bf5100a89e78: Pull complete
98f47fcaa52f: Pull complete
622a9dd8cfed: Pull complete
Digest: sha256:c18b9ace5dd1864674064dea03f7ff4e378e43b9ec57827853d0bd93953772df
Status: Downloaded newer image for httpd:latest
e3b8da676cc99af74e22bfd2dca833465095727e08253aece16bde650a524a1b
```

容器ID

请解析上图命令执行过程

运行一个容器 (2)

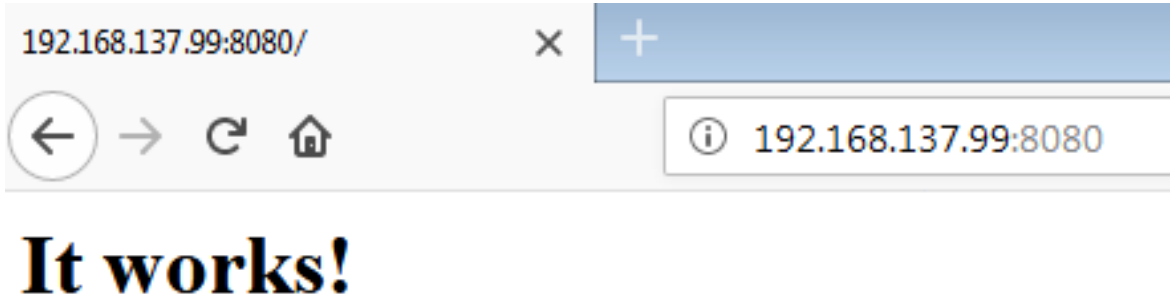
- 使用docker images查看下载的镜像。

```
[root@localhost ~]# docker images
REPOSITORY      TAG             IMAGE ID        CREATED         SIZE
httpd           latest          ee39f68eb241    2 weeks ago     154MB
```

- 使用docker ps命令查看容器运行状态。

```
[root@localhost ~]# docker ps
CONTAINER ID    IMAGE    COMMAND              CREATED        STATUS         PORTS                    NAMES
e3b8da676cc9    httpd    "httpd-foreground"   18 hours ago   Up 18 hours    0.0.0.0:8080->80/tcp     hungry_mcclintock
```

- 在浏览器上输入“宿主机IP:端口”，验证容器的可用性。



容器生命周期管理

- 使用docker stop命令停止一个容器。

```
[root@localhost ~]# docker stop e3b8da676cc9
e3b8da676cc9
```

- 使用docker ps -a命令可查看所有状态的容器。上一步中被stop的容器状态是Exited。

```
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE    COMMAND              CREATED         STATUS                       PORTS    NAMES
e3b8da676cc9   httpd    "httpd-foreground"   23 hours ago    Exited (137) 15 seconds ago          hungry_mcclintock
```

- 使用docker start命令启动一个容器。

```
[root@localhost ~]# docker start hungry_mcclintock
hungry_mcclintock
```

- 使用docker ps -a命令查看刚才被启动的容器，状态是Up。

```
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE    COMMAND              CREATED         STATUS          PORTS                   NAMES
e3b8da676cc9   httpd    "httpd-foreground"   23 hours ago    Up 14 seconds   0.0.0.0:8080->80/tcp    hungry_mcclintock
```

进入容器的方法

- 若要进入容器进行调试，有两种方法：
  - docker attach命令
    - docker attach命令直接进入已启动容器的命令终端，不会启动新的进程。
    - Usage：docker attach [OPTIONS] CONTAINER
  - docker exec命令
    - docker exec命令是在容器中打开新的终端。
    - Usage：docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

进入一个容器

- 使用docker attach命令进入一个容器。

```
[root@localhost ~]# docker run -d centos /bin/bash -c "while true; do sleep 1; echo Huawei; done"
b589b73d4bc382b77b39d9e751b5bd83c6e3bee884a23c71f1c0a0cc5fb92142
[root@localhost ~]# docker attach b589b73d4bc3
Huawei
Huawei
Huawei
```

- 使用docker exec命令进入同一个容器。

```
[root@localhost ~]# docker exec -it 98011f688cb0 bash
[root@98011f688cb0 /]# pwd
/
[root@98011f688cb0 /]# ls
anaconda-post.log  bin  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@98011f688cb0 /]# cat /etc/redhat-release
CentOS Linux release 7.6.1810 (Core)
[root@98011f688cb0 /]# ps ax
   PID TTY      STAT   TIME COMMAND
     1 ?        Ss     0:00 /bin/bash -c while true; do sleep 1; echo Huawei; done
   481 pts/0    Ss     0:00 bash
   512 ?        S      0:00 sleep 1
   513 pts/0    R+     0:00 ps ax
[root@98011f688cb0 /]# exit
exit
[root@localhost ~]#
```

知识小考

- docker kill、docker stop与docker pause命令的区别?

- Docker中响应用户API的是哪个组件?

实验&实训任务

- 实验任务
  - 请按照实验手册1.2部分完成容器基础操作实验。
- 实训任务
  - 请灵活使用本章节课程及实验手册中学到的知识，按照实验手册1.2.4章节完成容器基础操作实训任务。

思考题

1. docker容器的状态有以下哪些？（ ）

   A. created

   B. exited

   C. running

   D. paused

2. 处于exited状态的容器，会占用系统什么资源？

3. 删除所有终止状态容器的命令是什么？

本章总结

- 容器技术发展
- 容器核心价值
- 容器编排引擎
- OCI规范
- Docker架构
- 容器生命周期管理
- 进入一个容器

学习推荐

- Docker官网：https://www.docker.com/
- Docker项目：https://github.com/docker
- Docker中文社区：http://www.docker.org.cn/
- Kubernetes官网：https://kubernetes.io
- Kubernetes项目：https://github.com/kubernetes
- Kubernetes中文社区：https://www.kubernetes.org.cn/


谢 谢
www.huawei.com