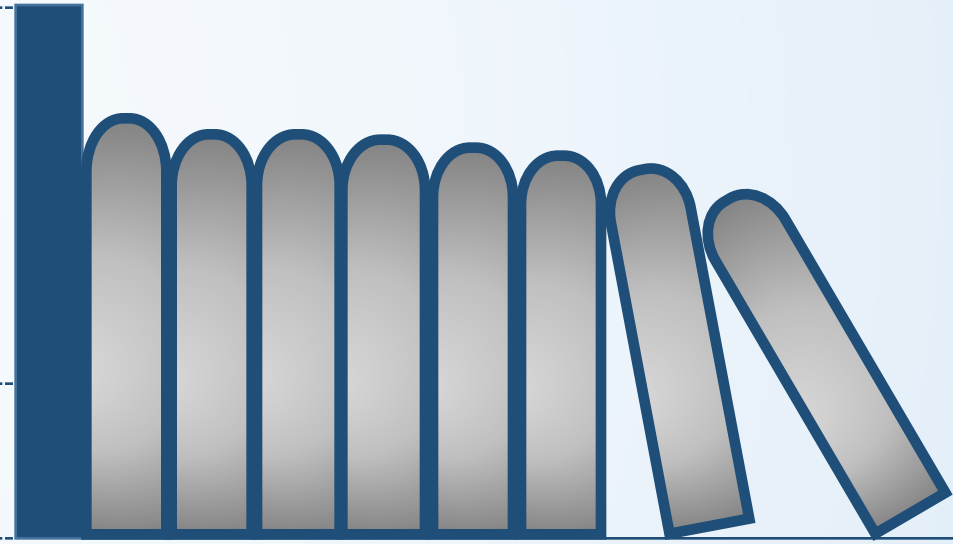
Docker基础

主讲人：宋小金



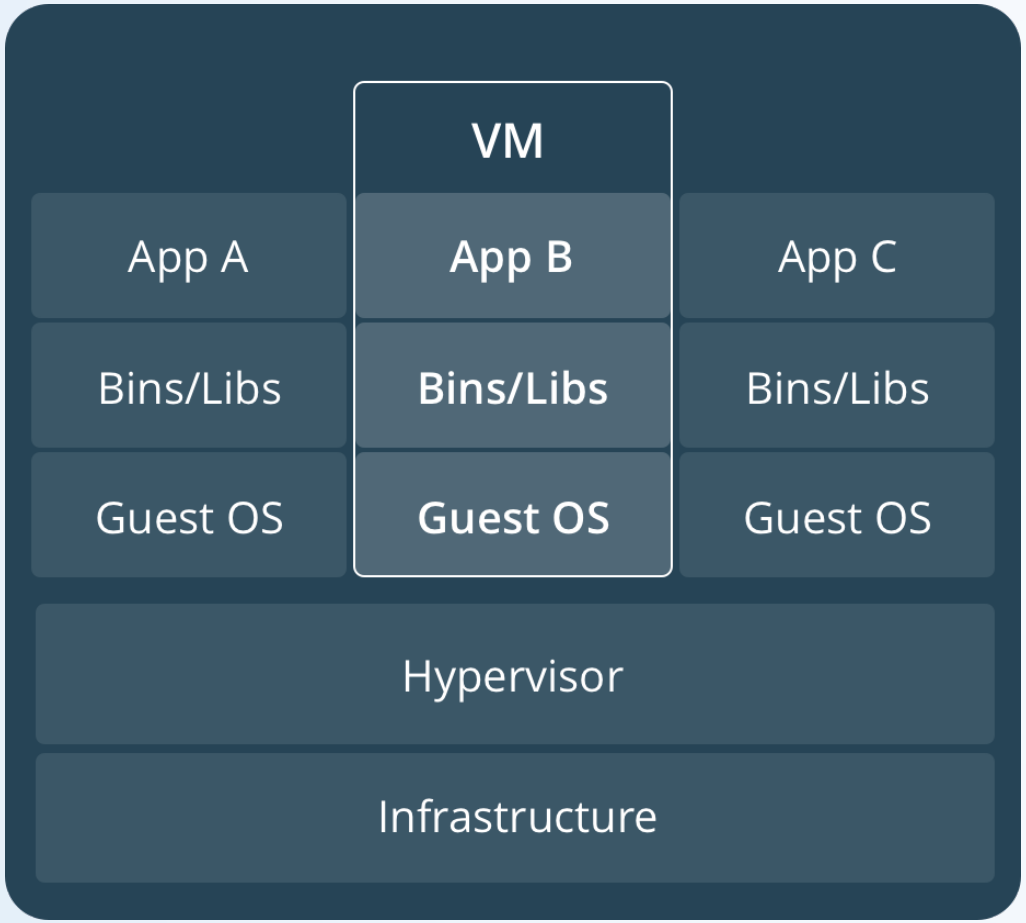
目录

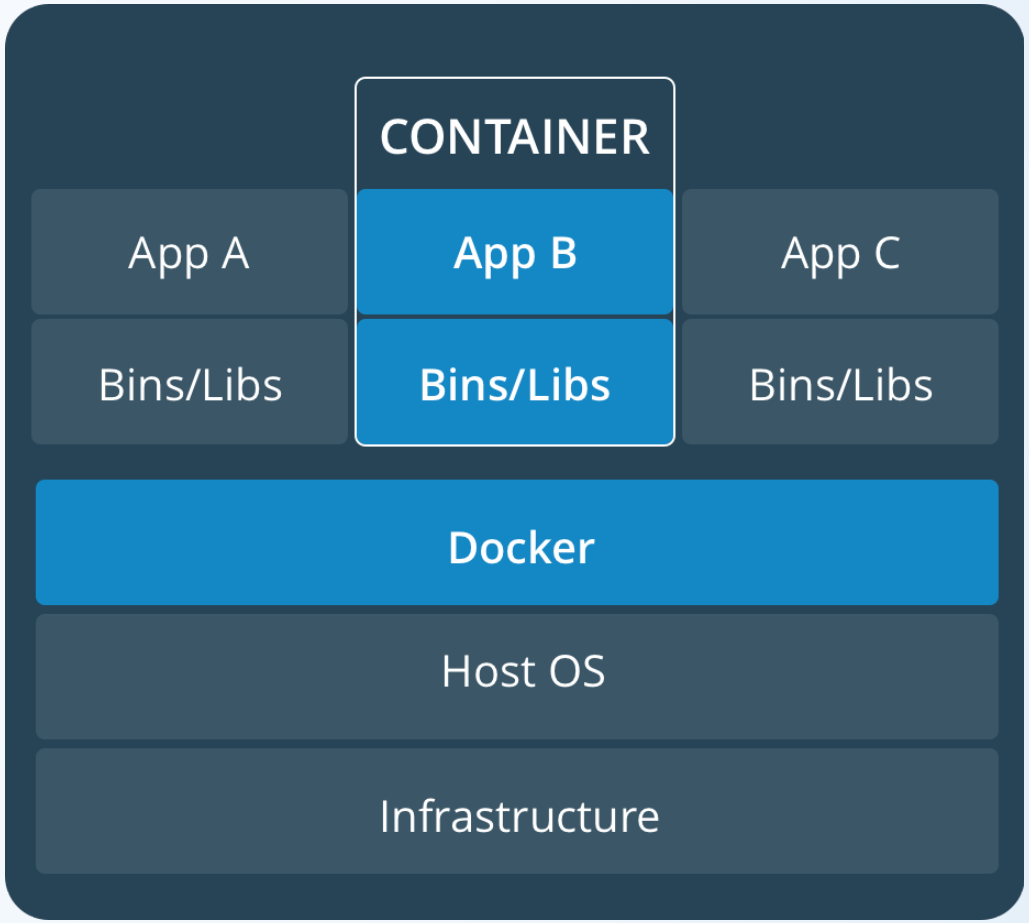
1 预期收获

- 对Docker有初步认识
- 了解容器与虚拟机差异
- 了解Docker组成
- 容器安装配置及实操


WORLD HISTORY
Physics
Chemistry
LAW

什么是容器


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure

物理机

虚拟机

容器

一栋楼一户人家，
独立地基，独立花园

一栋楼包含多套房，
一套房一户人家，
共享地基，共享花园，独立卫生间、厨房和宽带

一套房隔成多个小隔间(胶囊式公寓)，每个胶囊住一位租户，共享地基，共享花园，还共享卫生间、厨房和宽带

虚拟机 vs 容器

- 传统虚拟机技术通过Hypervisor层抽象底层基础设施资源，提供相互隔离的虚拟机，在其上运行一个完整的操作系统，在该系统上再运行所需的应用进程

- 容器是通过Linux内核技术对进程的资源，运行环境进行相对的限制和隔离，容器没有自己的内核，没有完整的操作系统，也没有对硬件虚拟化，所以更轻量化

- 虚拟机是为提供系统环境而生，容器则是为提供应用环境

性能对比

特性	虚拟机	容器
操作系统	非常广泛	Linux为主
隔离策略	Hypervisor	Namespace
启动时间	分钟级	秒级
资源损耗	5-15%	0-5%
镜像存储	GB-TB	KB-MB
集群规模	数百	近万

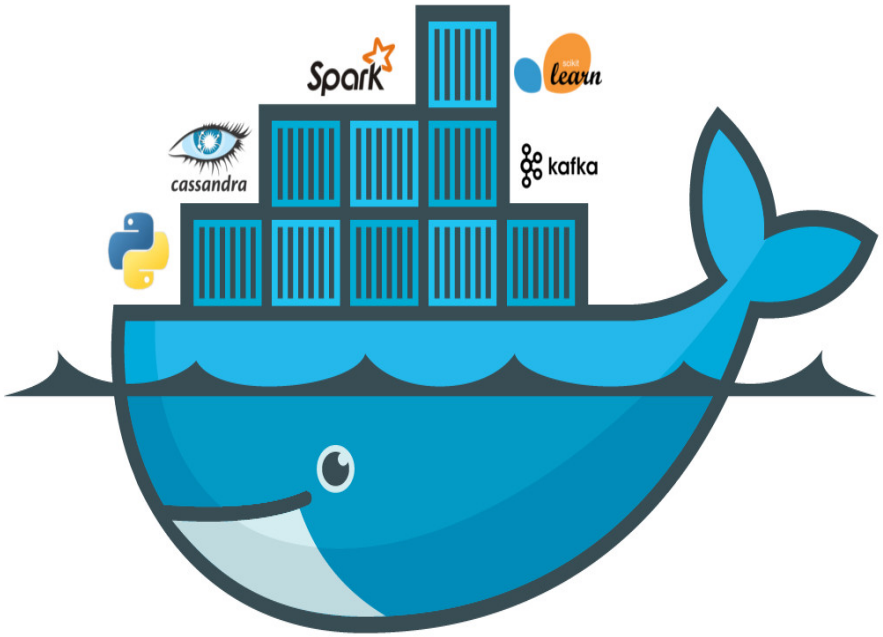
为什么要用容器

1. 更高效的利用系统资源 --- 性能损耗少，调度颗粒细
2. 更快速的启动时间 --- 进程形式，秒级启动
3. 一致的运行环境 --- 镜像
4. 持续交付部署和弹性 --- 资源池统一
5. 更轻松的迁移 云原生目标，开箱即用，比IaaS更轻松
6. 更轻松的维护和扩展

总结：任何IT技术兴起都可从三个维度来分析：提高稳定性，提升效率，节省资源成本，基于云原生来说就是编排

Docker简介

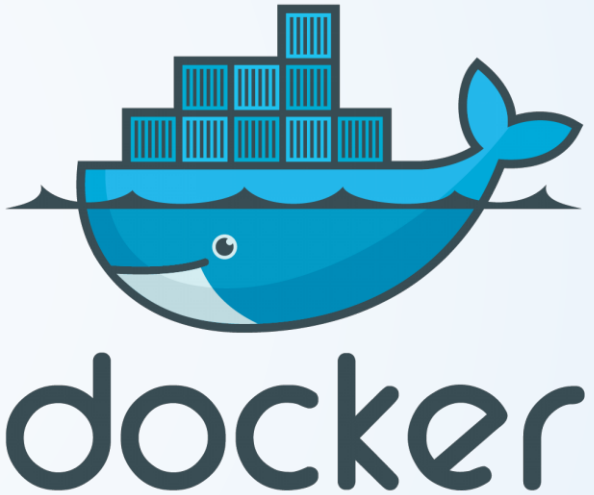
- DotCloud公司创始人 Solomon Hykes 在法国期间发起一个公司内部项目
- 基于*Go语言*并遵循Apache 2.0协议开源
- 基于 *Linux 内核*的 cgroup，namespace等技术对进程进行隔离。
- Docker 让开发者可以打包他们的*应用以及依赖包*到一个可移植的容器中，然后发布到任何流行的 Linux 机器上。
- 容器是完全使用*沙箱机制*，相互之间不会有任何接口。



Docker容器

- Run applications securely isolated in a container.
- Packaged with all its dependencies and libraries.
- Build, Ship, and Run Any App, Anywhere.

思考：Docker是容器，但容器不仅只是Docker，那为什么Docker兴起了，之前有么？Docker以后又有那些类似产品？



Docker容器

Q1：Docker以前有容器么

- Docker以前早有容器，甚至Docker的早期版本就是基于LXC（Linux Container）开发的
- Google的Borg系统有类似的产品，Oracle也有

Q2：为什么Docker火了

- 时机好，来的是时候，有了微服务架构
- 镜像特性，即装箱技术，参照右图Logo
- Build, Ship, and Run Any App, Anywhere

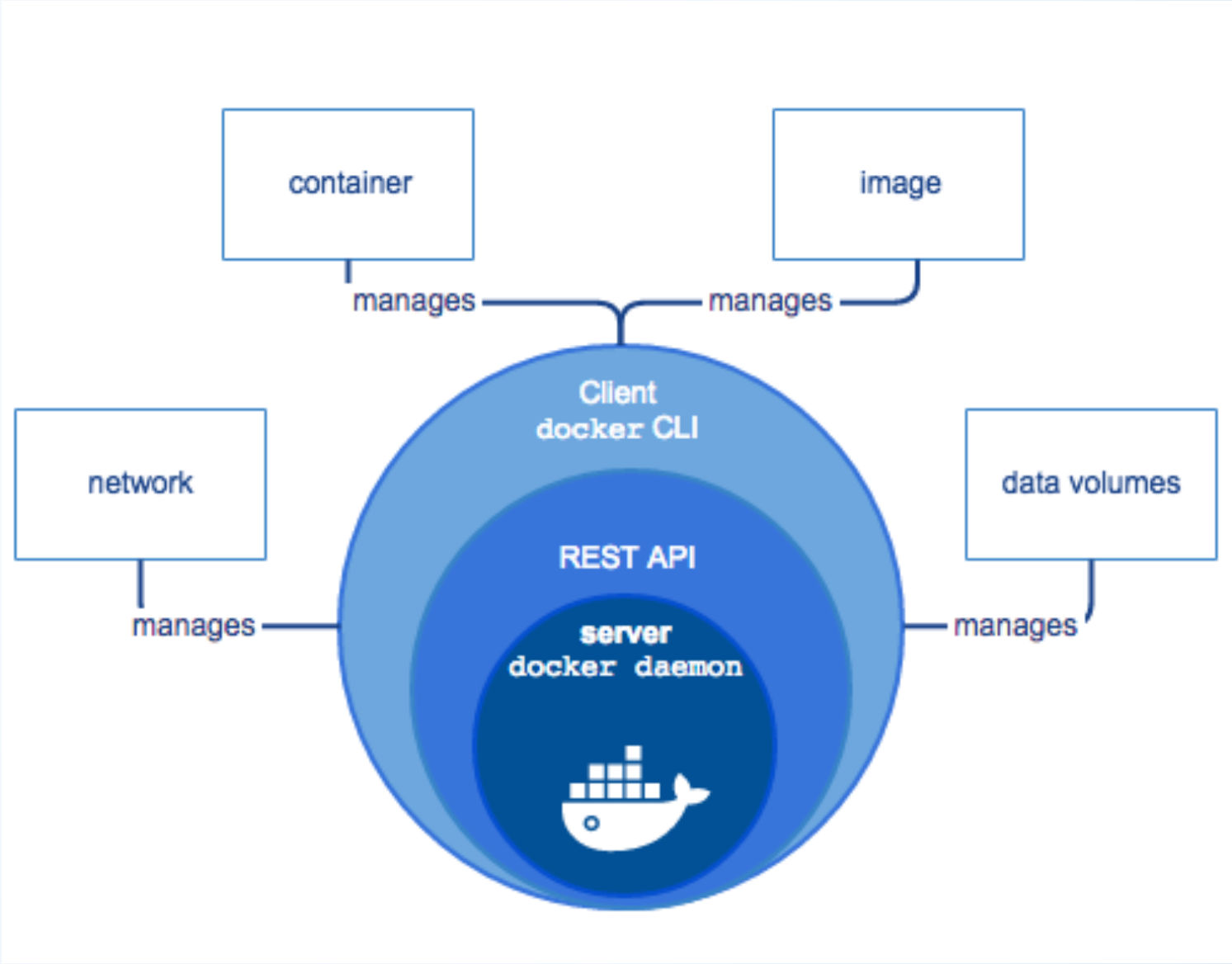
Q3：Docker之后有什么类似产品

- Kata Container【OpenStack出品】
- Gvisor【Google出品】
- Rkt【CoreOS出品】

Docker 概览

Docker采用C/S架构，主要由以下几个组件组成：

- Docker Server: 是通过*dockerd*以守护进程运行的程序
- CLI：*命令行*工具*docker*
- REST API：通过REST API与Docker Daemon进行交互，比如CLI或者直接调用REST API



Docker 架构

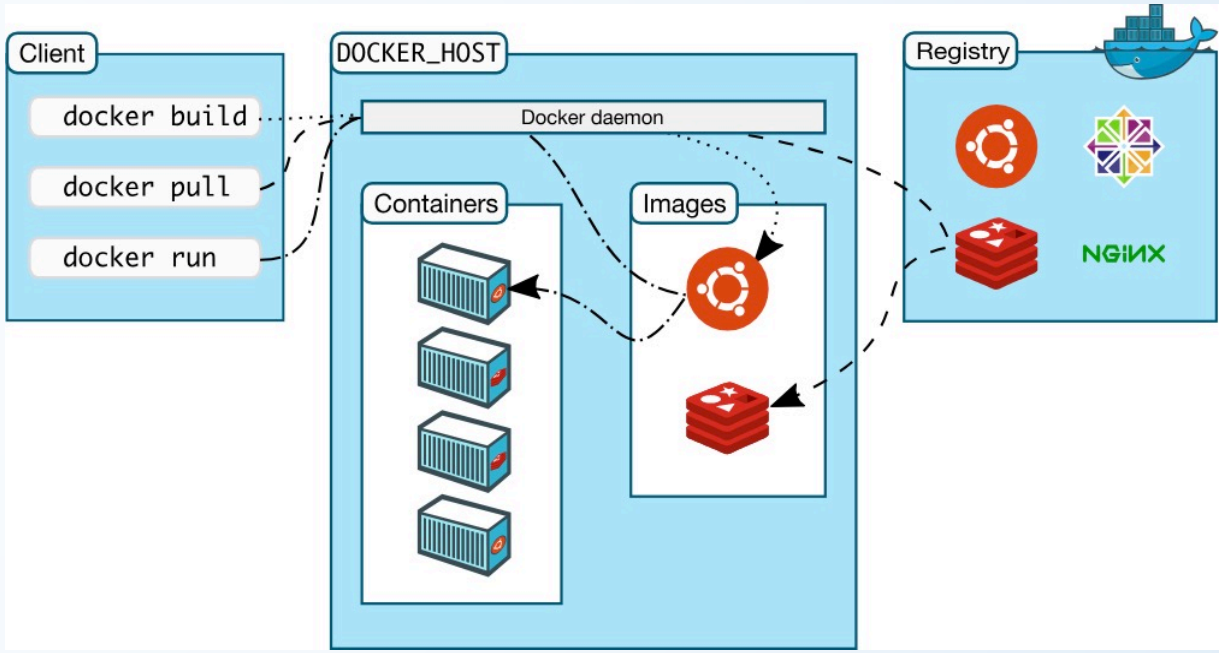
Docker采用C/S架构，主要有以下几个组件：

- Docker Clinet与Docker Daemon可*运行在同一个节点或不同节点上*，Docker Clint通过*REST API或UNIX socket*与Docker Daemon通信。

- Docker Daemon：处理Docker API请求，并对*镜像、容器网络、存储卷*等对象进行管理。

- Docker Client：用于用户与*Docker Daemon交互*。例如，*docker run* 命令，Docker Client会把命令通过Docker API发送给dockerd，然后dockerd来负责执行。

- Docker Registry：*镜像存储*，如DockerHub



自建Docker Registry

服务器bjo-ep-018.dev.fwmrm.net上执行如下命令：

```
$ docker run -d -p 5000:5000  -name registry registry:2
```

对Alpine镜像打新的tag：

```
$ docker tag alpine bjo-ep-018.dev.fwmrm.net:5000/alpine
```

配置insecure-registry：docker daemon启动参数加上“-insecure-registry bjo-ep-018.dev.fwmrm.net:5000”，并重启docker daemon

上传镜像：

```
$ docker push bjo-ep-018.dev.fwmrm.net:5000/alpine
```

The push refers to a repository [bjo-ep-018.dev.fwmrm.net:5000/alpine]
cd7100a72410: Pushed
latest: digest: sha256:cac7fac5a6b0f450e85865fdb468e06acef0bc6e81645a7130fcda871f1c505a size:528

举例1：Docker Clinet远程与Docker Daemon通信。

```
ts1@bjoadsbuild01.dev.fwmrm.net:~ · 01:42 PM Fri May 04 ·
!3761 $ docker -H bjo-ep-018.dev.fwmrm.net:2375  run -itd alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
ff3a5c916c92: Pull complete
Digest: sha256:7df6db5aa61ae9480f52f0b3a06a140ab98d427f86d8d5de0bedab9b8df6b1c0
Status: Downloaded newer image for alpine:latest
45085d08c816ee4d10947416ab9c981ae2dfcaa97e0aef2b8dd933045c1a1d33
```

```
ts1@bjo-ep-018.dev.fwmrm.net:~ · 01:43 PM Fri May 04 ·
!119 $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
                    NAMES
45085d08c816        alpine              "/bin/sh"           5 seconds ago       Up 4 seconds
                    vigilant_jones

ts1@bjo-ep-018.dev.fwmrm.net:~ · 01:46 PM Fri May 04 ·
!120 $
```

举例2：通过API查看容器信息

```
$ curl http://bjo-ep-018.dev.fwmrm.net:2375/containers/json
[
    {
        "Id":"45085d08c816ee4d10947416ab9c981ae2dfcaa97e0aef2b8dd933045c1a1d33",
        "Names":[
            "/vigilant_jones"
        ],
        "Image":"alpine",
        "ImageID":"sha256:3fd9065eaf02feaf94d68376da52541925650b81698c53c6824d92ff63f98353",
        "Command":"/bin/sh",
        "Created":1525441572,
        "Ports":[

        ],
        "Labels":{

        },
        "State":"running",
        "Status":"Up 6 minutes",
        "HostConfig":{
```

Docker的商业化道路

2017年年初，docker公司将原先的docker项目改名为moby，并创建了Docker-CE和Docker-EE。

三者关系：

- moby是继承了原先的docker的项目，是社区维护的的开源项目，谁都可以在moby的基础打造自己的容器产品
- docker-ce是docker公司维护的开源项目，是一个基于moby项目的免费的容器产品
- docker-ee是docker公司维护的闭源产品，是docker公司的商业产品

公司	开源版本	社区版本	企业版本
Redhat	Fedora	CentOS	RHEL
Docker	Moby	Docker-CE	Docker-EE

Docker的安装与配置

参照附件：01　Docker的安装与配置.txt

当你运行一个容器的时候

$docker run -it nginx /bin/bash

按次序，Docker Engine做：

1. Pull nginx镜像
2. 创建一个新的容器分配文件系统，挂载一个读写层
3. 创建网络命名空间，分配一个网络/桥接口
4. 设置一个IP地址
5. 执行一个指定的进程【/usr/bin/nginx】
6. 捕获和获取应用的输出

Docker容器实操演示

参照附件：02　Docker容器实操演示.txt

课程回顾

已学知识要点

- *了解Docker与传虚拟机的区别*
- *了解Docker组成与架构*
- *一个容器的基本操作：创建、查看与删除*