Linux服务器构建与运维管理

第10章：使用Docker实现容器

阮晓龙

13938213680 / ruanxiaolong@hactcm.edu.cn

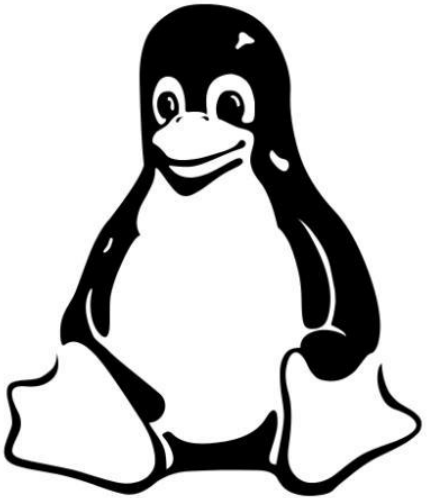
https://internet.hactcm.edu.cn
http://www.51xueweb.cn

河南中医药大学信息管理与信息系统教研室
河南中医药大学信息技术学院互联网技术教学团队
河南中医药大学医疗健康信息工程技术研究所

2022.10

提纲

- 容器技术
  - 认识容器技术
  - Docker
  - 容器与虚拟化
- 使用Docker实现容器
  - 安装Docker
  - 使用Docker实现LAMP
- 管理Docker
  - 使用Docker Compose管理Docker
  - 使用cAdvisor监控Docker性能
  - 使用第三方工具管理Docker
    - 可视化管理工具：Portainer、DockStation、Docker Desktop
    - 终端UI管理工具：Lazydocker、Docui

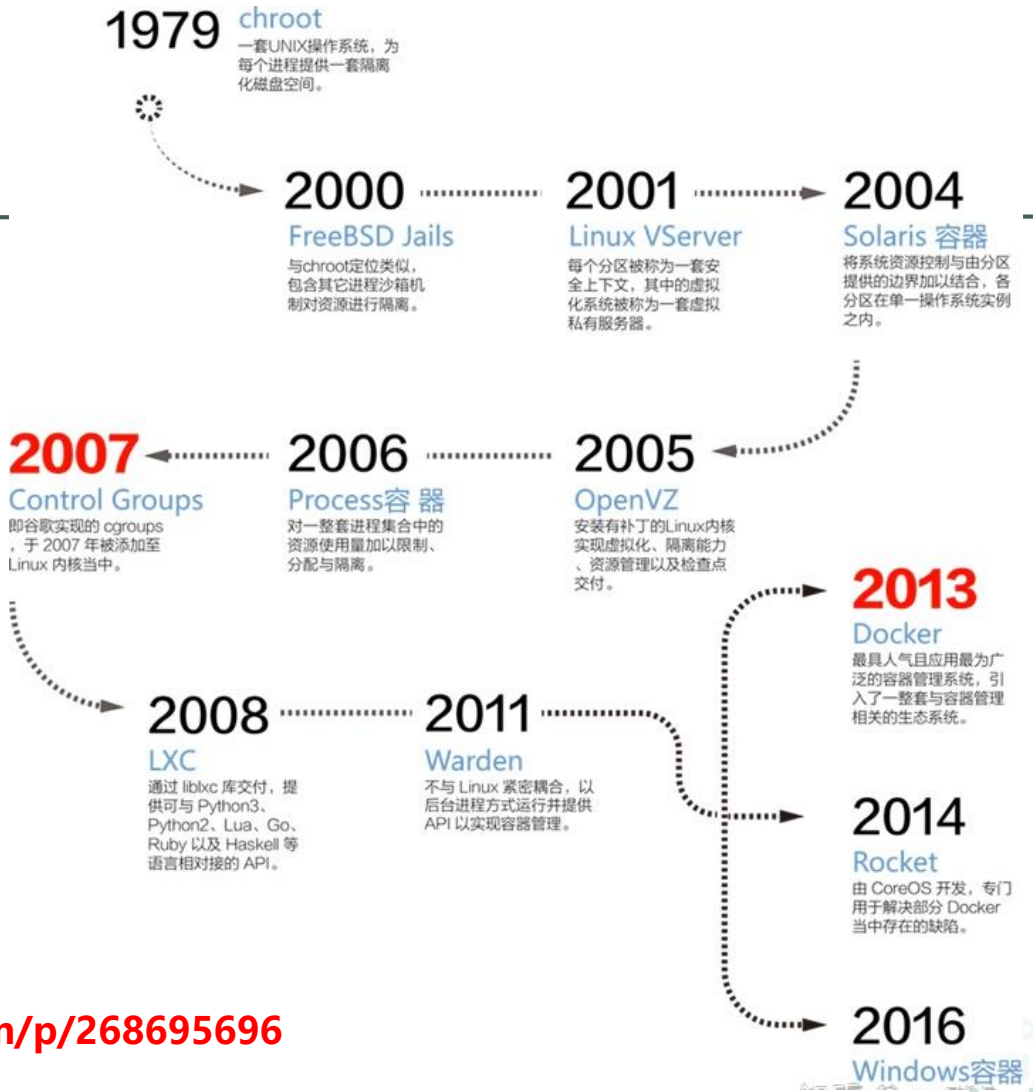


1.容器技术

- 容器是一种标准化的概念，其特点是规格统一，并且可层层堆叠。
- 在IT领域，容器名称为Linux Container（简称LXC），是一种操作系统层面的虚拟化技术，使用容器技术可将应用程序打包成标准的单元，便于开发、交付与部署。
- 容器的主要特点。
  - 容器是轻量级的可执行独立软件包，包含应用程序运行所需的所有内容，如代码、运行环境、系统工具、系统库与设置等。
  - 容器适用于基于Linux和Windows的应用程序，在任何环境中都能够始终如一地运行。
  - 容器赋予了应用程序独立性，使其免受外在环境差异的影响，有助于减少相同基础设施上运行不同应用程序时的冲突。

1.容器技术

- 容器的发展历史



知乎
https://zhuanlan.zhihu.com/p/268695696

1.容器技术

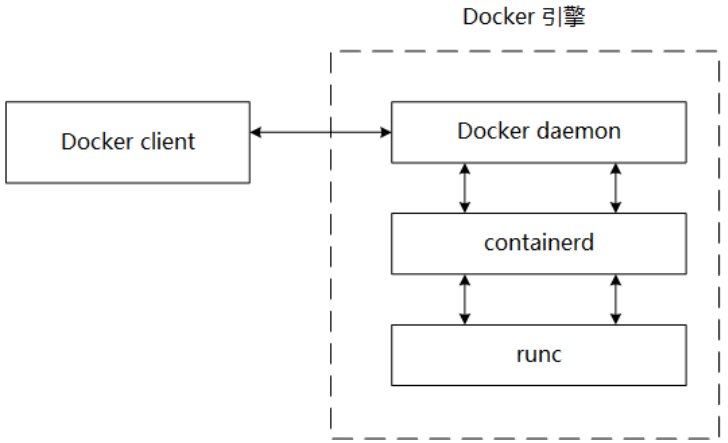
- LXC提供了对命令空间（Namespace）和资源控制组（CGroup）等Linux基础工具的操作能力，是基于Linux内核的容器虚拟化技术。
- LXC可以有效地将操作系统管理的资源划分到独立的组中，在共享操作系统底层资源的基础上，让应用程序独立运行。
  - 旧版本的Docker软件依托LXC实现，但由于LXC是基于Linux的，不易实现跨平台，Docker公司开发了名为Libconainer的工具用于替代LXC。
  - Libconainer是与平台无关的工具，可基于不同的内核为Docker软件上层提供容器交互功能。

1.容器技术

- Docker是基于Go语言实现的开源容器项目，其官方定义Docker为以Docker容器为资源分割和调度的基本单位，封装整个软件运行时的环境，为开发者和系统管理员设计，用于构建、发布、运行分布式应用的平台。
- Docker是一个跨平台的、可移植并简单易用的容器解决方案。
  - 目标是实现轻量级的操作系统虚拟化解决方案，通过对应用的封装、分发、部署、运行生命周期的管理，达到应用组件"一次封装，到处运行"的目的。
  - 目前已形成围绕Docker容器的生态体系。
  - Docker的官方网站为：https://www.docker.com。

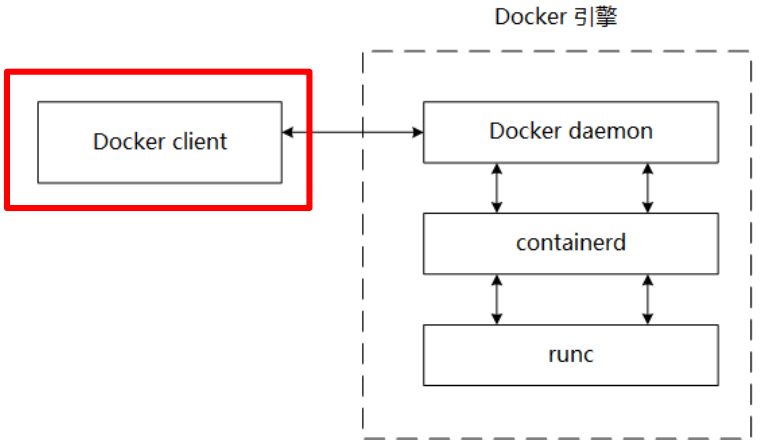
1.容器技术

- Docker引擎是用于运行和编排容器的基础设施工具，是运行容器的核心运行环境，相当于VMware体系中的ESXi。
  - 其他Docker公司或者第三方公司的产品都是围绕Docker引擎进行开发和集成的。
  - 构成Docker引擎的组件有Docker client、Docker daemon、containerd和runc。



1.容器技术

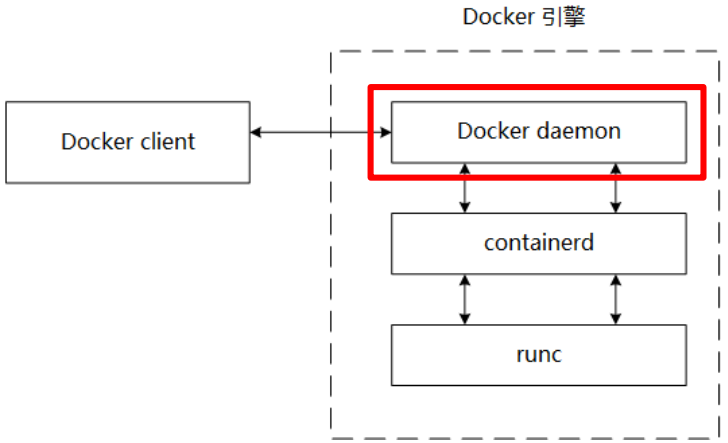
- Docker引擎是用于运行和编排容器的基础设施工具，是运行容器的核心运行环境，相当于VMware体系中的ESXi。
  - 其他Docker公司或者第三方公司的产品都是围绕Docker引擎进行开发和集成的。
  - 构成Docker引擎的组件有Docker client、Docker daemon、containerd和runc。



- Docker client是Docker用户与Docker交互的主要方式。当执行docker run之类的命令时，Docker client将通过Docker API的方式发送命令给Docker daemon。
- Docker client可以与多个Docker daemon通信。

1.容器技术

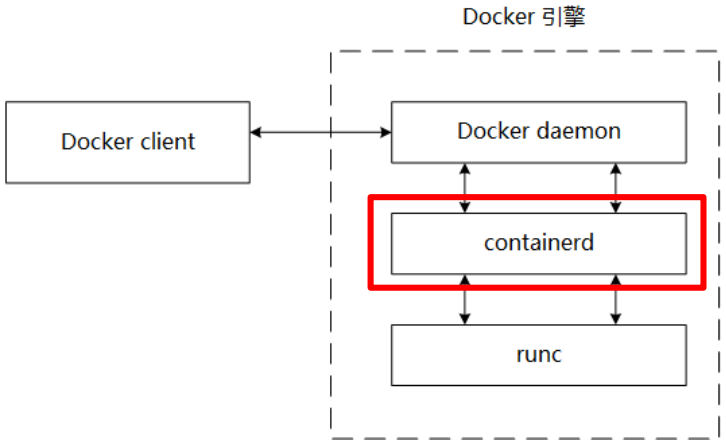
- Docker引擎是用于运行和编排容器的基础设施工具，是运行容器的核心运行环境，相当于VMware体系中的ESXi。
  - 其他Docker公司或者第三方公司的产品都是围绕Docker引擎进行开发和集成的。
  - 构成Docker引擎的组件有Docker client、Docker daemon、containerd和runc。



- Docker daemon是Docker的守护进程，用于侦听Docker API请求并管理Docker对象。
- 例如镜像、容器、网络和存储卷。

1.容器技术

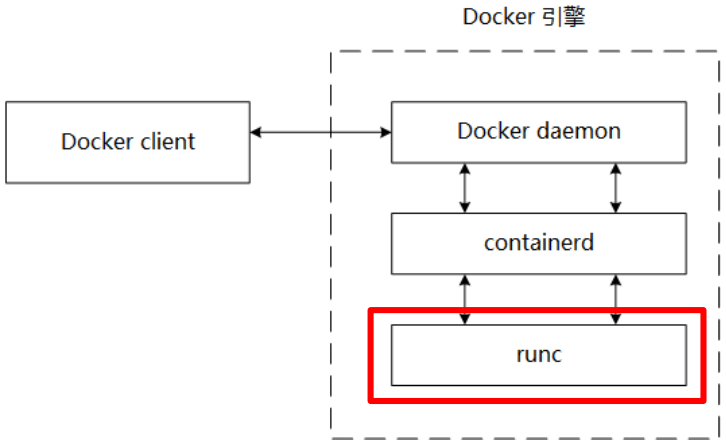
- Docker引擎是用于运行和编排容器的基础设施工具，是运行容器的核心运行环境，相当于VMware体系中的ESXi。
  - 其他Docker公司或者第三方公司的产品都是围绕Docker引擎进行开发和集成的。
  - 构成Docker引擎的组件有Docker client、Docker daemon、containerd和runc。



- containerd的主要任务是容器的生命周期管理，可在宿主机中管理完整的容器生命周期。
- 例如容器镜像的传输和存储、容器的执行和管理、存储、网络等。

1.容器技术

- Docker引擎是用于运行和编排容器的基础设施工具，是运行容器的核心运行环境，相当于VMware体系中的ESXi。
  - 其他Docker公司或者第三方公司的产品都是围绕Docker引擎进行开发和集成的。
  - 构成Docker引擎的组件有Docker client、Docker daemon、containerd和runc。



- 为了维护容器生态，Docker公司与Core公司共同成立了一个旨在管理容器标准的委员会（简称为OCI）。目前，OCI已经发布两份规范：镜像规范和运行时规范。
- Docker引擎中的runc是OCI规定的容器运行时规范的实现，其实质上是一个轻量级的、针对Libcontainer进行封装的命令行交互工具，仅用于创建容器。

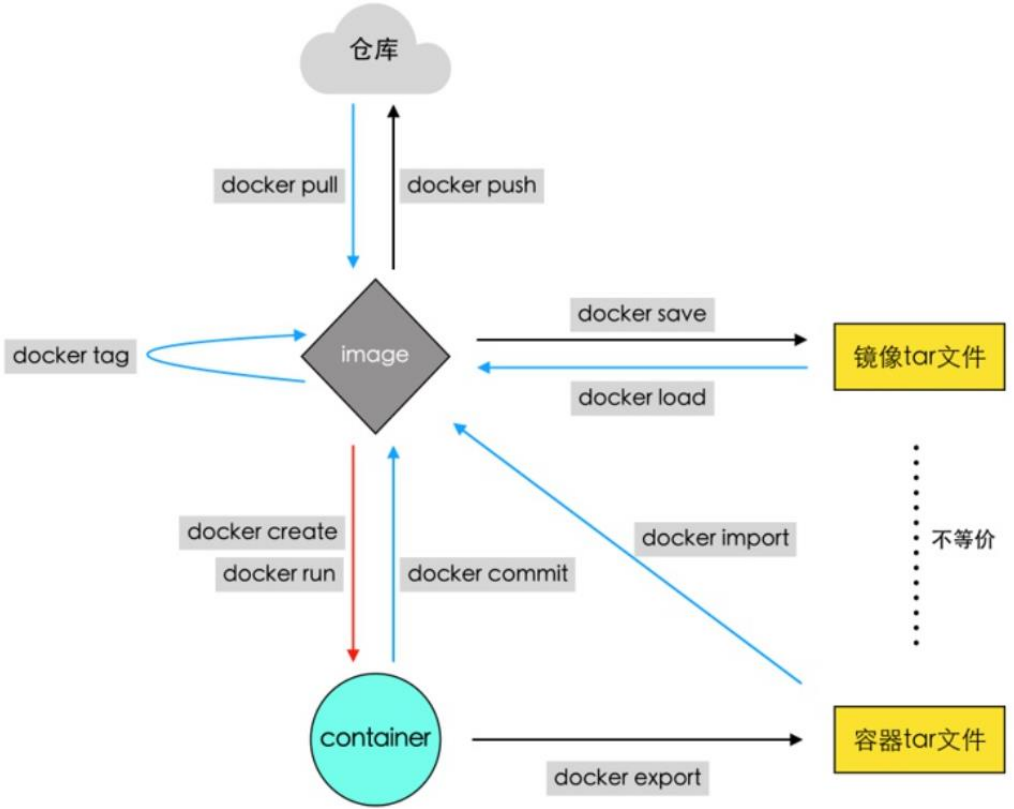
1.容器技术

- Docker包含三个核心概念：镜像（Image）、容器（Container）、仓库（Repository）。
- 理解Docker核心概念有助于理解Docker的整个生命周期。
  - 镜像是一个只读的文件系统
    - 镜像的核心是一个精简的操作系统，同时包含软件运行所必须的文件和依赖包。镜像由多个镜像层构成，每次叠加后，从外部来看镜像就是一个独立的对象。
    - 镜像是分层存储的架构。每个Docker镜像实际是由多层文件系统联合组成。镜像构建时，会一层层构建，前一层是后一层的基础。每一层构建完就不会再发生改变，后一层上的任何改变只发生在自己所在的层。因为容器设计的初衷就是快速和小巧，因此镜像通常比较小，例如，Docker官方镜像Alpine Linux仅有4MB左右。
  - 容器是镜像运行的实例
  - 仓库是集中存放镜像文件的地方

1.容器技术

- Docker包含三个核心概念：镜像（Image）、容器（Container）、仓库（Repository）。
- 理解Docker核心概念有助于理解Docker的整个生命周期。
  - 镜像是一个只读的文件系统
  - 容器是镜像运行的实例
    - 容器是镜像运行时的实体，可以被创建、启动、停止、删除、暂停等。
    - 容器启动时将在容器的最上层创建一个可写层，其生存周期和容器一样，容器消亡时，容器可写层也随之消亡。任何保存于容器可写层的信息都会随容器删除而丢失。
  - 仓库是集中存放镜像文件的地方
    - 仓库是集中存储、分发镜像的服务，分为公开和私有两种。
    - Docker默认使用的是公开仓库服务，默认选择的公开仓库服务是官方提供的Docker Hub，地址是https://hub.docker.com。

1.容器技术



docker的生命周期中，镜像和容器是最重要的两部分。

- 镜像是文件，是一个只读的模板，一个独立的文件系统，里面包含运行容器所需的数据，可以用来创建新的容器。
- 容器是基于镜像创建的进程，容器中的进程依赖于镜像中的文件，容器具有写的功能，可以根据需要改写里面的软件、配置等，并可以保存为新的镜像。
- 如果用import方法生成，则是一个完全新的镜像。
- 如果用commit方法生成，则新镜像与原来的镜像之间存在着继承关系。

1.容器技术

- Docker的主要优势如下。
  - 轻量
    - 在一台机器上运行的多个Docker容器可以共享这台机器的操作系统内核；它们能够迅速启动，只需占用很少的计算和内存资源。
  - 标准
    - Docker容器基于开放式标准，能够在所有主流Linux版本、Microsoft Windows以及包括VM、裸机服务器和云在内的任何基础设施上运行。
  - 安全
    - Docker赋予应用的隔离性不仅限于彼此隔离，还独立于底层的基础设施。Docker默认提供强隔离，因此应用出现问题，也只是单个容器的问题，不会波及到整台机器。

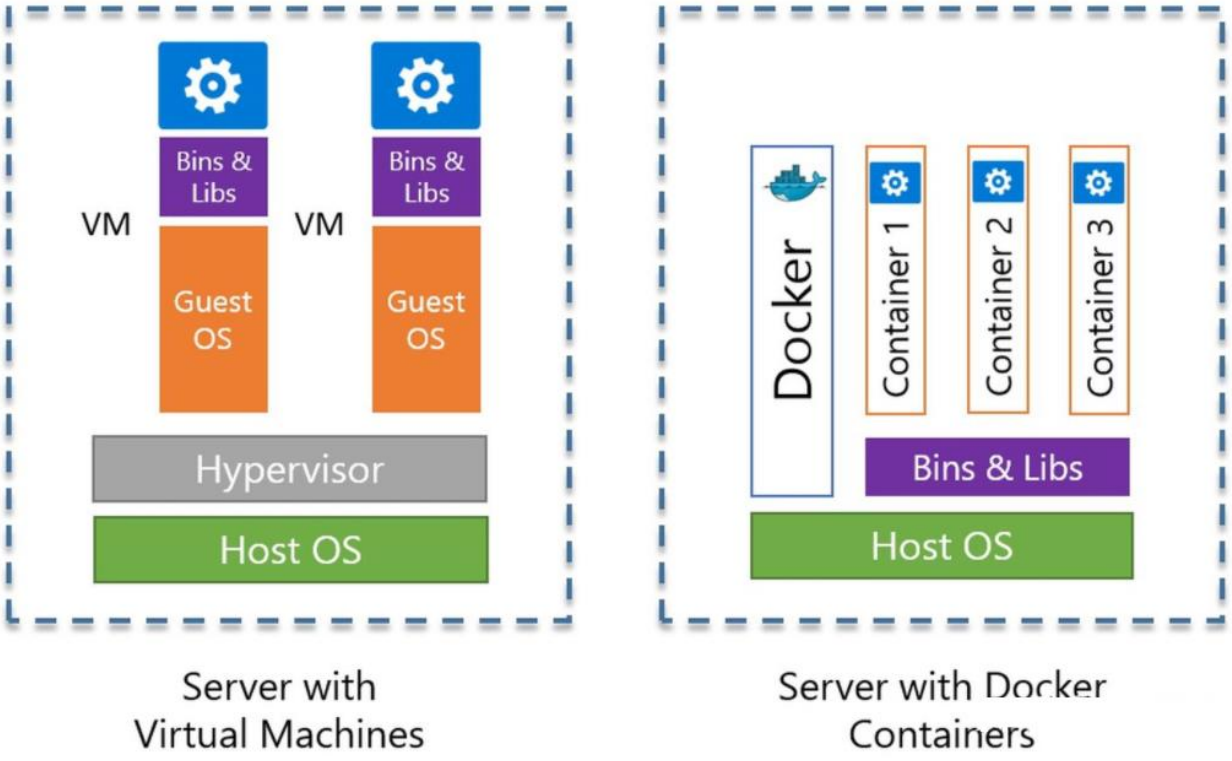
1.容器技术

- 使用Docker可以实现开发人员的开发环境、测试人员的测试环境、运维人员的生产环境的整体一致性，因此Docker的主要应用场景如下。
  - Web应用的自动化打包和发布。
  - 自动化测试和持续集成、发布。
  - 在服务型环境中部署和调整数据库或其他的后台应用。

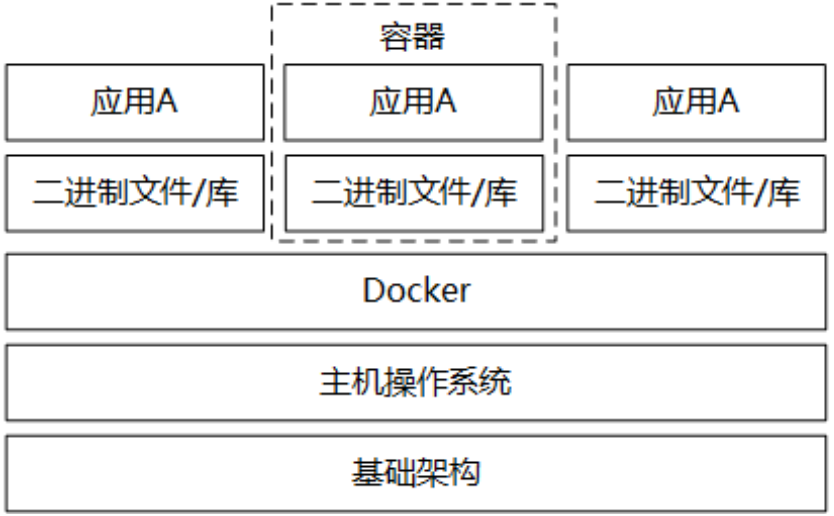
1.容器技术

- Docker容器和虚拟机有很多相似的地方，比如资源隔离、分配优势，但其功能并不相同。
  - Docker容器虚拟化的是操作系统，虚拟机虚拟化的是硬件。虚拟机是将硬件物理资源划分为虚拟资源，属于硬件虚拟化。容器将操作系统资源划分为虚拟资源，属于操作系统虚拟化。
  - 虚拟机是虚拟出一套硬件后，在其上运行一个完整操作系统，在该系统上再运行所需应用进程；而Docker容器内的应用进程则直接运行于宿主机的内核，Docker容器没有自己的内核，而且也没有进行硬件虚拟，相对来讲，Docker容器比虚拟机更加简洁高效。
  - Docker技术与虚拟机技术有着不同的使用场景。虚拟机更擅长于彻底隔离整个运行环境，例如，云服务提供商通常采用虚拟机技术隔离不同的用户。Docker技术通常用于隔离不同的应用，例如前端、后端以及数据库的部署。

1.容器技术


Bins & Libs
VM
Bins & Libs
VM
Guest OS
Guest OS
Hypervisor
Host OS
Server with Virtual Machines
Docker
Container 1
Container 2
Container 3
Bins & Libs
Host OS
Server with Docker Containers

1.容器技术


容器
应用A
应用A
应用A
二进制文件/库
二进制文件/库
二进制文件/库
Docker
主机操作系统
基础架构


虚拟机
应用A
应用B
应用C
二进制文件/库
二进制文件/库
二进制文件/库
客户机操作系统
客户机操作系统
客户机操作系统
管理程序
基础架构

docker

Products Developers Pricing Blog About us

Sign In Get Started

DockerCon LIVE 2021 Watch Now!

Catch up on the latest product announcements, interviews, sessions, panels, and 45+ sessions. Now available on-demand.

Watch Now!

aws DATADOG MIRANTIS Red Hat vmware

WHAT'S NEW

Docker Expands Trusted Content Offerings for Developers

New collaborations with AWS, Datadog, Mirantis, Red Hat, VMware and other industry leaders expand access to trusted application building blocks to more than eight million registered Docker developers.

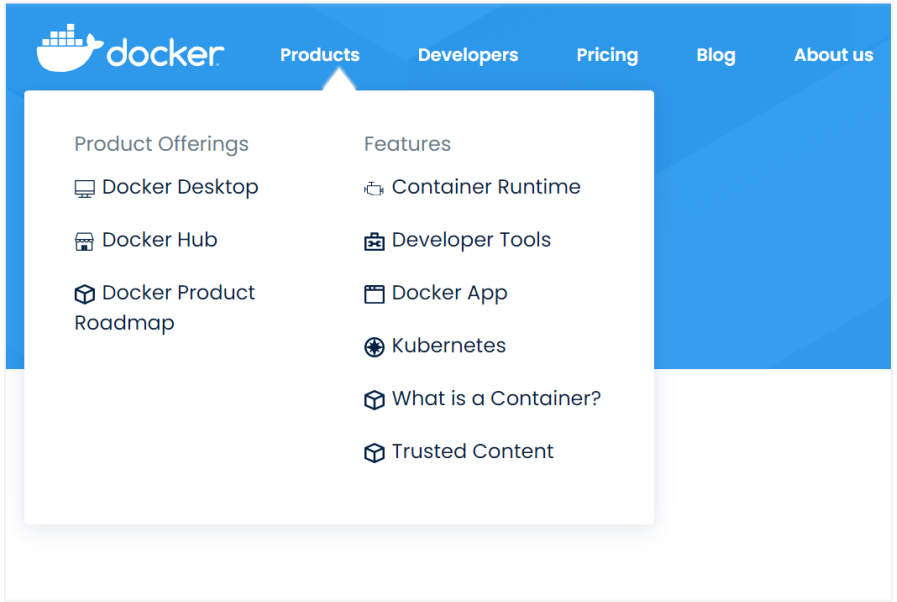
→ Learn More!

Accelerate how you build, share, and run modern applications.

11 million +
developers

7 million +
applications

13 billion +
monthly image downloads



FAQs

Q: Is everything on the roadmap?

A: Almost everything related to features, we might keep the odd feature a surprise if we really aren't sure about it. We also won't be tracking all of our bugs here, we have separate repos for these:

- Docker Hub
- Docker Engine
- Docker Desktop Windows
- Docker Desktop Mac

Q: What do the roadmap categories mean?

- Shipped - Hopefully you are enjoying it! Give us feedback on how it is working!
- Almost There - We are applying the finishing touches. Things in this bucket you can expect to be shipped within 2-4 weeks.
- We're Writing the Code - Actively in development, we are trying to get this out to you in a good state as soon as we can.
- Investigating - We're thinking about it. This might mean we're still designing, or thinking through how this might work. This is a great phase to send how you want to see something implemented! We'd love to see your usecase or design ideas here.

Q: Why are there no dates on your roadmap?

A: Because we know things change and we want the room to do the right thing by fixing security issues as they come up or helping people out where they need. This means we might have to change our priorities and don't want to let people down.

Q: How can I provide feedback or ask for more information?

A: Please open an issue in this repo! If the issue is a bug or security issue, please follow the separate instructions above.

Q: How can I request a feature be added to the roadmap?

A: Please open an issue! You can read about how to contribute here. Community submitted issues will be tagged "Proposed" and will be reviewed by the team.

Q: Will you accept a pull request?

A: We haven't worked out how pull requests should work for a public roadmap page, but we will take all PRs very seriously and review for inclusion. Read about contributing. We review these each week.

Pricing & Subscriptions

Choose a plan that's right for you.

Free

FOR EVERYBODY

- Unlimited public repositories
- Docker Desktop continuously updated
- Docker Desktop includes Docker Engine and Kubernetes
- Limited container image requests
- Two-factor authentication

$0

Signup for Free

DEVELOPER FAVORITE

Pro

FOR INDIVIDUALS

- Everything in Free
- Unlimited private repositories
- Unlimited container image requests
- 2 parallel builds
- 300 monthly Hub image vulnerability scans
- Premium customer support for Desktop and Hub

$5 per user/month

With annual plan

Buy Now

Team

FOR ORGANIZATIONS

- Everything in Pro
- Unlimited teams
- Unlimited monthly Hub image vulnerability scans
- 3 parallel builds per org
- Role-based access control
- Audit log
- First five members is $25, $7 per additional team member

$7 per user/month

With annual plan

Buy Now

Large

FOR ORGANIZATIONS

- Everything in Team
- Whitelist service up to 20 IPs
- Invoicing available
- Minimum 500 users
- $84/team seat per year
- Governed by Terms of Service

Contact Sales

Open Source Plans

Docker is proud to support the Open Source community. Qualifying OSS projects can get no-cost Docker accounts to support their contributors and end users.

Apply Now →

Verified Publishers

The Docker Verified Publisher Program helps you improve your customer experience, build customer trust, increase awareness, and much more.

Learn More →

2.使用Docker实现容器

任务1：安装Docker

任务2：使用Docker实现LAMP

2.使用Docker实现容器

任务1：安装Docker

- 步骤1：创建虚拟机并完成CentOS的安装
- 步骤2：完成虚拟机的主机配置、网络配置及通信测试
- 步骤3：验证是否满足Docker安装要求
- 步骤4：查询并删除旧版本Docker软件
- 步骤5：设置Docker的yum仓库源

2.使用Docker实现容器

任务1：安装Docker

- 步骤6：通过在线方式安装Docker软件
- 步骤7：启动docker服务
- 步骤8：查看Docker运行信息
- 步骤9：配置docker服务为开机自启动
- 步骤10：验证是Docker软件是否安装成功

操作视频 / 现场演示

- 任务1：安装Docker
  - 任务目标：
    - 实现在线安装Docker。
    - 实现docker服务管理。
    - 实现docker服务状态查看。

2.使用Docker实现容器

任务2：使用Docker实现LAMP

- 步骤1：创建PHP文件
- 步骤2：准备Docker网络实现容器互联
- 步骤3：创建mariadb容器实现数据库服务
- 步骤4：创建php容器实现PHP程序运行
- 步骤5：验证LAMP是否部署成功
- 步骤6：查看容器运行状态

操作视频 / 现场演示

- 任务2：使用Docker实现LAMP
  - 任务目标：
    - 基于Docker发布LAMP。

3.管理Docker

Docker Desktop

Docker Compose

cAdvisor

Other

3.管理Docker

- Docker Compose是Docker官方提供的管理工具，用来实现对多个容器的快速管理。
- Compose is a tool for defining and running multi-container Docker applications.
  - With Compose, you use a YAML file to configure your application's services.
  - Then, with a single command, you create and start all the services from your configuration.

A `docker-compose.yml` looks like this:

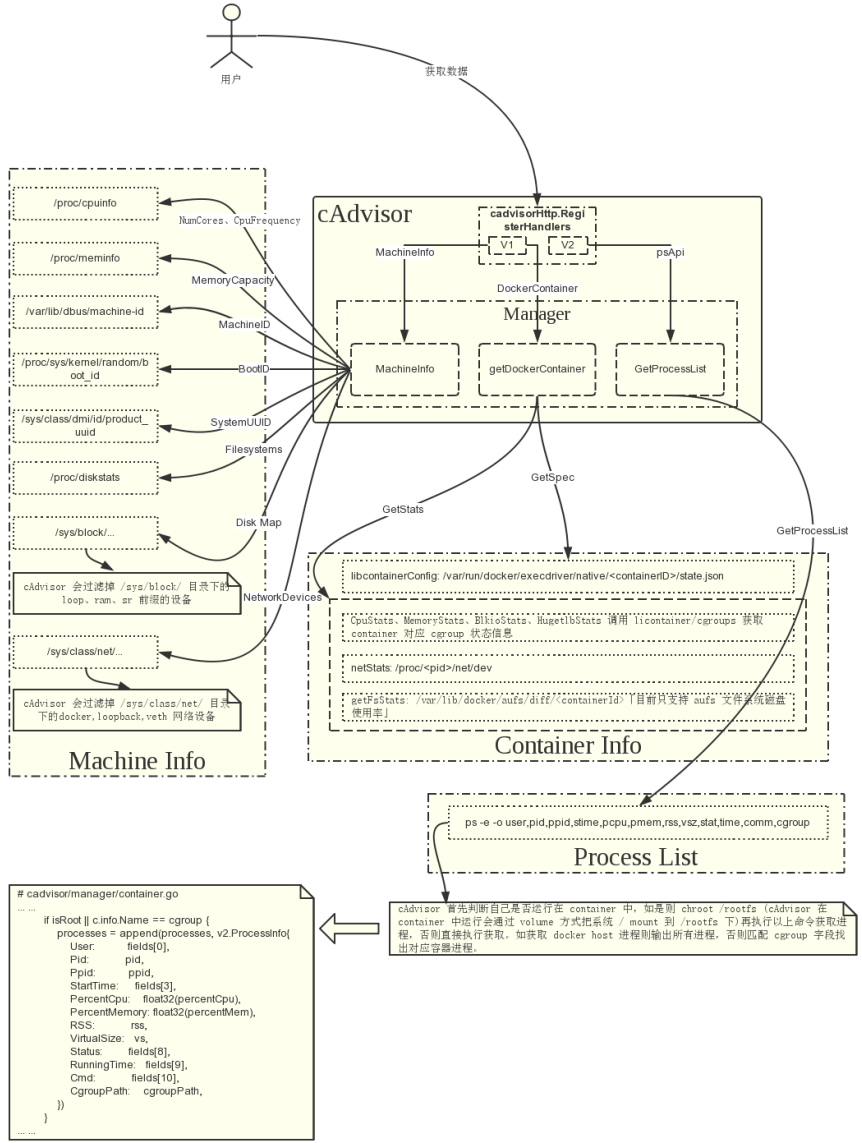
```yaml
version: "3.9"  # optional since v1.27.0
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

3.管理Docker

- cAdvisor是开源的Docker容器监控工具，支持对安装Docker的宿主机、Docker自身的实时监控和性能数据采集。
- cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers.
  - It is a running daemon that collects, aggregates, processes, and exports information about running containers.
  - Specifically, for each container it keeps resource isolation parameters, historical resource usage, histograms of complete historical resource usage and network statistics. This data is exported by container and machine-wide.

cAdvisor

https://github.com/google/cadvisor



物理机

虚拟机

容　器

Build, Ship and Run

Build once，Run anywhere


智能运维课程体系


棒棒堂