Linux 操作系统
(基于 openEuler)

第10章：使用Docker实现容器服务

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技术学院智能医疗教研室
河南中医药大学医疗健康信息工程技术研究所

2025.8

提纲

- 理解容器
  - 容器
  - Docker
- 通过Docker实现容器应用
  - 安装Docker
  - 基于Docker实现PostgreSQL数据库服务
  - 监控Docker
- 使用Docker Compose部署容器应用
  - 了解Docker Compose
  - 使用Docker Compose部署Drupal
- 深入理解物理机、虚拟机和容器
  - Bare metal、VMs、Containers

1. 理解容器

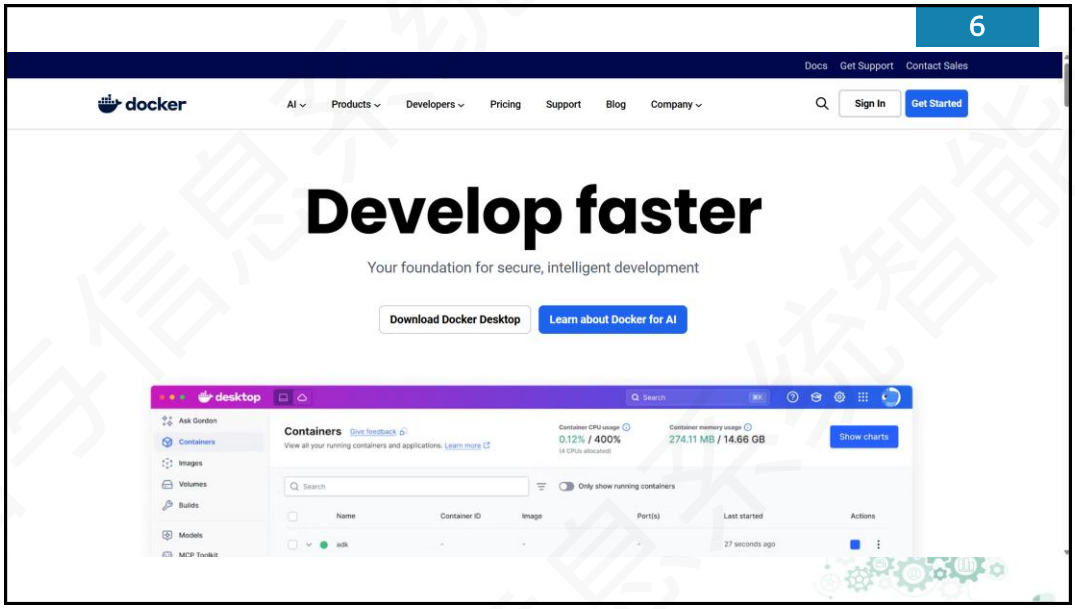
- 容器是一种标准化的概念，其特点是规格统一，并且可层层堆叠。
  - 在IT领域，容器名称为Linux Container（简称“LXC”），是一种操作系统层面的虚拟化技术。
  - 使用容器技术可将应用程序打包成标准的单元，便于开发、交付与部署。
- 容器的主要特点
  - 容器是轻量级的可执行独立软件包，包含应用程序运行所需的所有内容，如代码、运行环境、系统工具、系统库与设置等。
  - 容器适用于基于Linux和Windows的应用程序，在任何环境中都能够始终如一地运行。
  - 容器赋予了应用程序的独立性，使其免受外在环境差异的影响，有助于减少相同基础设施上运行不同应用程序时的冲突。

1. 理解容器

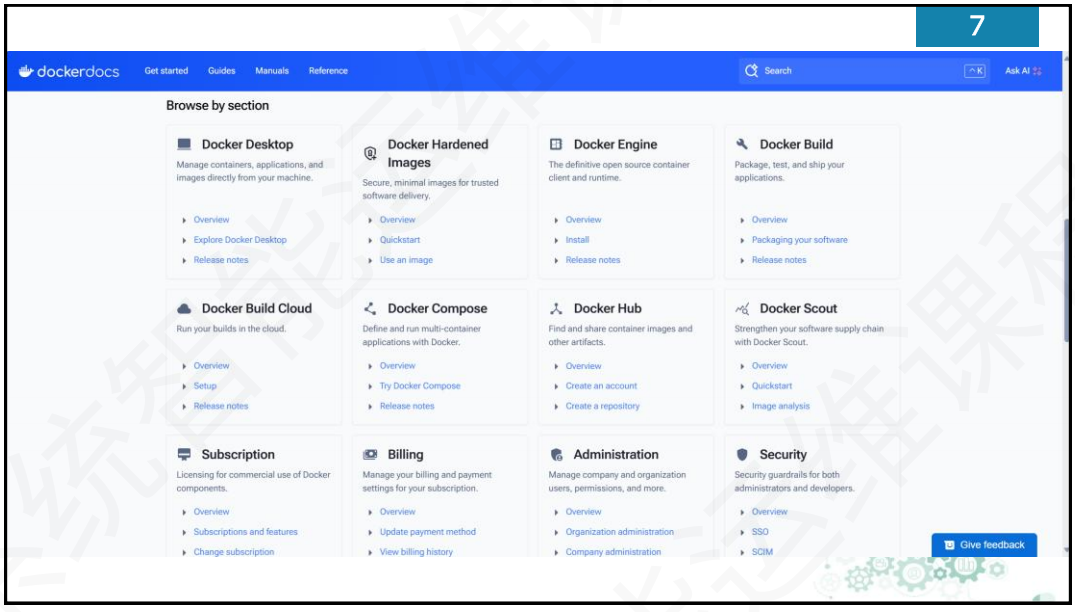
- LXC
  - LXC提供了对命令空间（Namespace）和资源控制组（CGroup）等Linux基础工具的操作能力，是基于Linux内核的容器虚拟化技术。
  - LXC可以有效地将操作系统管理的资源划分到独立的组中，在共享操作系统底层资源的基础上，让应用程序独立运行。
  - 旧版本的Docker软件依托LXC实现，但由于LXC是基于Linux的，不易实现跨平台，故Docker公司开发了名为Libcontainer的工具用于替代LXC。
  - Libcontainer是与平台无关的工具，可基于不同的内核为Docker软件上层提供容器交互功能。

1. 理解容器

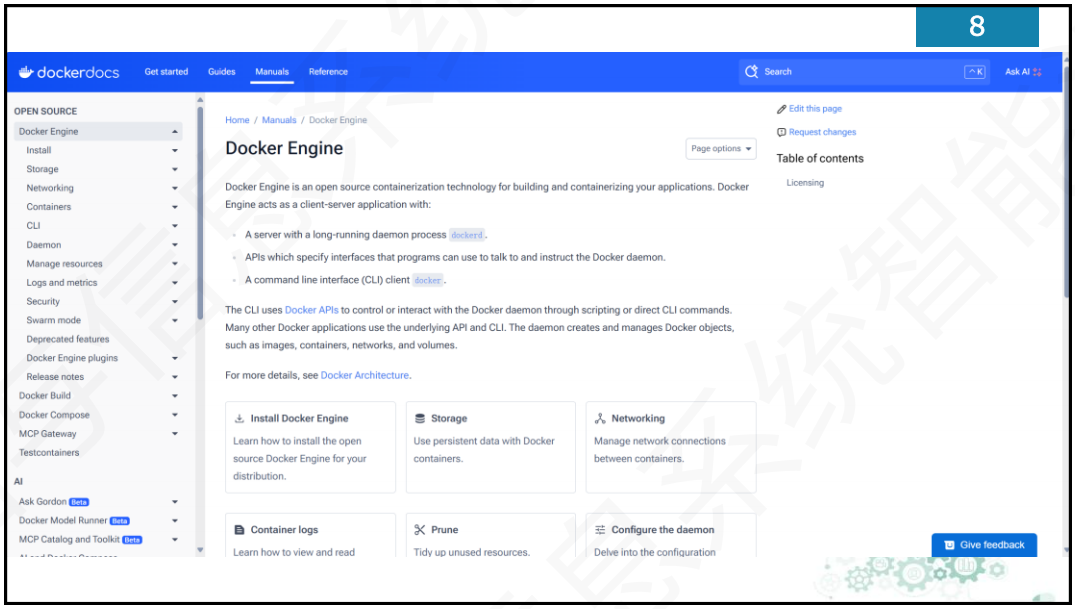
- Docker是基于Go语言实现的开源容器项目，其官方定义Docker为以Docker容器为资源分割和调度的基本单位，封装整个软件运行时的环境，为开发者和系统管理员设计，用于构建、发布、运行分布式应用的平台。
- Docker是一个跨平台的、可移植并简单易用的容器解决方案。
  - 其目标是实现轻量级的操作系统虚拟化解决方案，通过对应用的封装、分发、部署、运行生命周期的管理，达到应用组件“一次封装，到处运行”的目的。
- 目前已形成围绕Docker容器的生态体系。
  - Docker的官方网站为：https://www.docker.com。



7

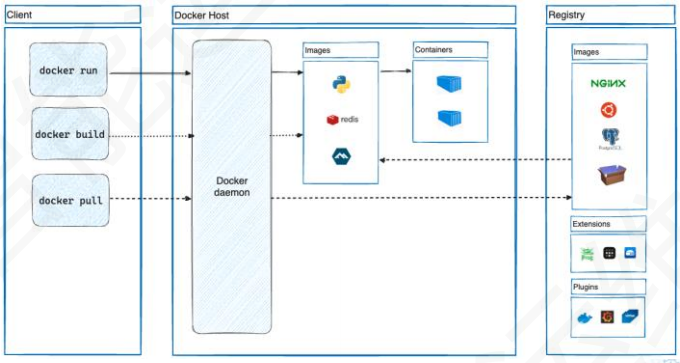

dockerdocs
Get started
Guides
Manuals
Reference
Search
Ask AI
Browse by section
Docker Desktop
Manage containers, applications, and images directly from your machine.
Overview
Explore Docker Desktop
Release notes
Docker Hardened Images
Secure, minimal images for trusted software delivery.
Overview
Quickstart
Use an image
Docker Engine
The definitive open source container client and runtime.
Overview
Install
Release notes
Docker Build
Package, test, and ship your applications.
Overview
Packaging your software
Release notes
Docker Build Cloud
Run your builds in the cloud.
Overview
Setup
Release notes
Docker Compose
Define and run multi-container applications with Docker.
Overview
Try Docker Compose
Release notes
Docker Hub
Find and share container images and other artifacts.
Overview
Create an account
Create a repository
Docker Scout
Strengthen your software supply chain with Docker Scout.
Overview
Quickstart
Image analysis
Subscription
Licensing for commercial use of Docker components.
Overview
Subscriptions and features
Change subscription
Billing
Manage your billing and payment settings for your subscription.
Overview
Update payment method
View billing history
Administration
Manage company and organization users, permissions, and more.
Overview
Organization administration
Company administration
Security
Security guardrails for both administrators and developers.
Overview
SSO
SCIM
Give feedback

8


dockerdocs
Get started
Guides
Manuals
Reference
Search
Ask AI
OPEN SOURCE
Docker Engine
Install
Storage
Networking
Containers
CLI
Daemon
Manage resources
Logs and metrics
Security
Swarm mode
Deprecated features
Docker Engine plugins
Release notes
Docker Build
Docker Compose
MCP Gateway
Testcontainers
AI
Ask Gordon Beta
Docker Model Runner Beta
MCP Catalog and Toolkit Beta
Home / Manuals / Docker Engine
Docker Engine
Page options
Edit this page
Request changes
Table of contents
Licensing
Docker Engine is an open source containerization technology for building and containerizing your applications. Docker Engine acts as a client-server application with:
A server with a long-running daemon process dockerd.
APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
A command line interface (CLI) client docker.
The CLI uses Docker APIs to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI. The daemon creates and manages Docker objects, such as images, containers, networks, and volumes.
For more details, see Docker Architecture.
Install Docker Engine
Learn how to install the open source Docker Engine for your distribution.
Storage
Use persistent data with Docker containers.
Networking
Manage network connections between containers.
Container logs
Learn how to view and read
Prune
Tidy up unused resources.
Configure the daemon
Delve into the configuration
Give feedback

1. 理解容器

- Docker架构

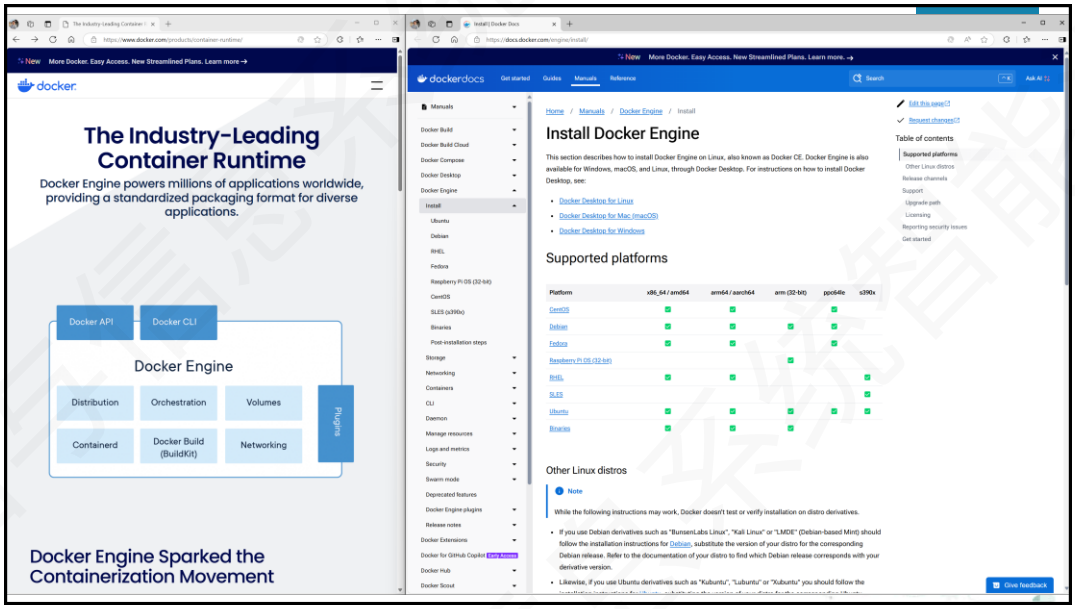


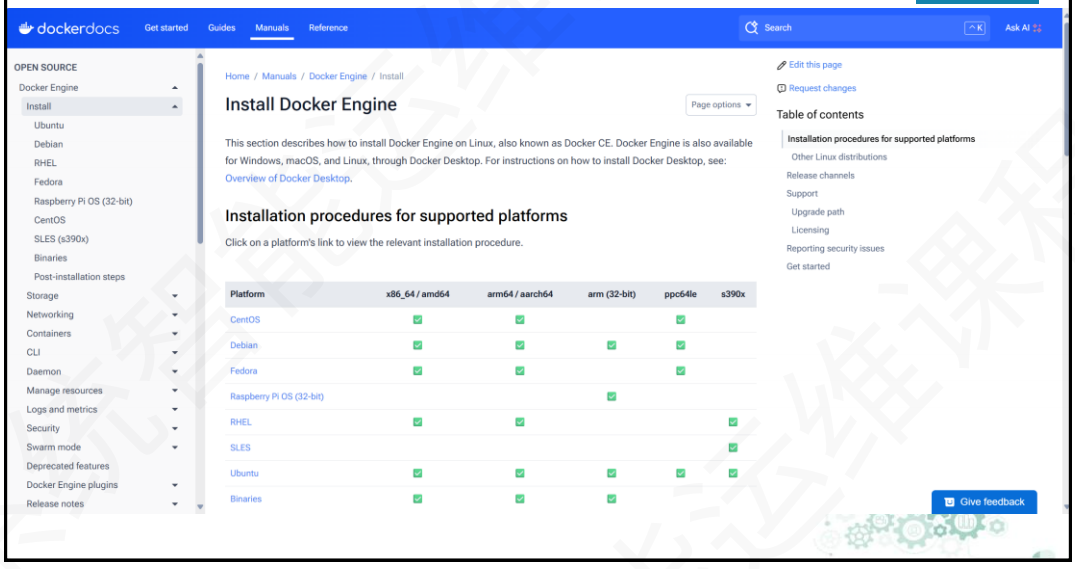
1. 理解容器

- Docker架构
  - Docker daemon：Docker守护进程（dockerd）
    - 用于监听Docker API请求并管理Docker对象，例如镜像、容器、网络和存储卷。
    - Docker daemon还可与其他守护程序通信以管理Docker服务。
  - Docker client：Docker客户端（docker）
    - 用户与Docker交互的主要方式，当使用诸如docker run之类的命令时，Docker client将发送命令到Docker daemon，由其执行。
    - Docker客户端可与多个Docker daemon通信。
  - Docker registries：Docker仓库
    - 用于存储Docker镜像。
    - Docker Hub（https://hub.docker.com）是Docker官方提供的镜像仓库，当使用docker pull之类的命令时，Docker会从Docker仓库中获取镜像，当使用docker push命令时，Docker会将本地镜像推送至Docker仓库中。

1. 理解容器

- Docker Engine
  - Docker Engine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，其中包括以下组件。
    - 持续运行的守护进程（dockerd）
    - 可与守护进程（dockerd）通信的API
    - Docker命令行客户端（cli）
  - CLI使用Docker API通过脚本或直接CLI命令来控制Docker守护程序或与之交互。
    - 其他Docker应用程序使用底层API和CLI。
    - 守护程序创建和管理Docker对象，例如图像、容器、网络和卷。

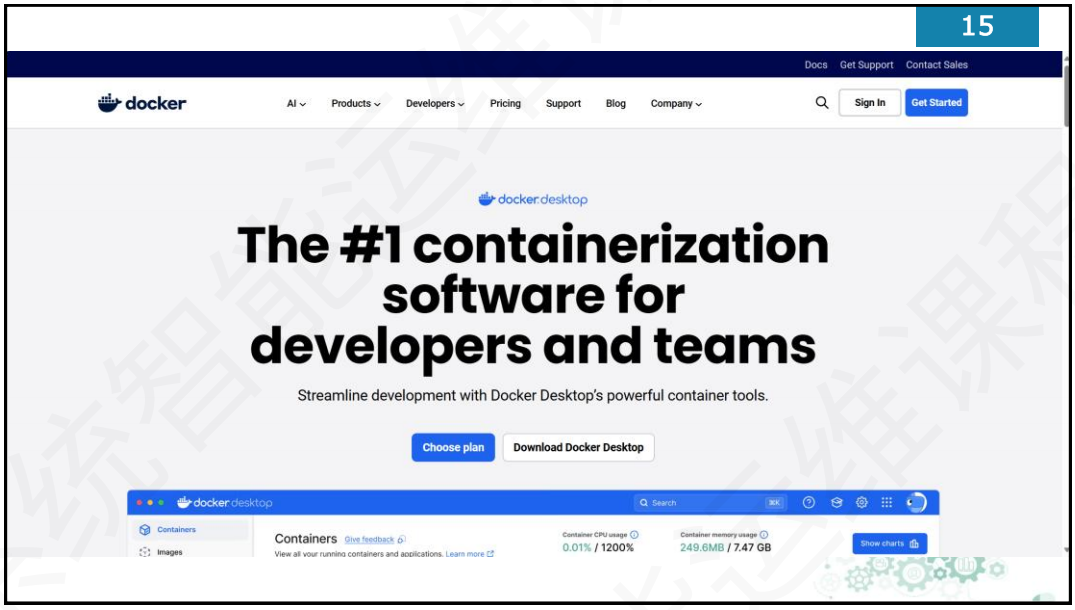




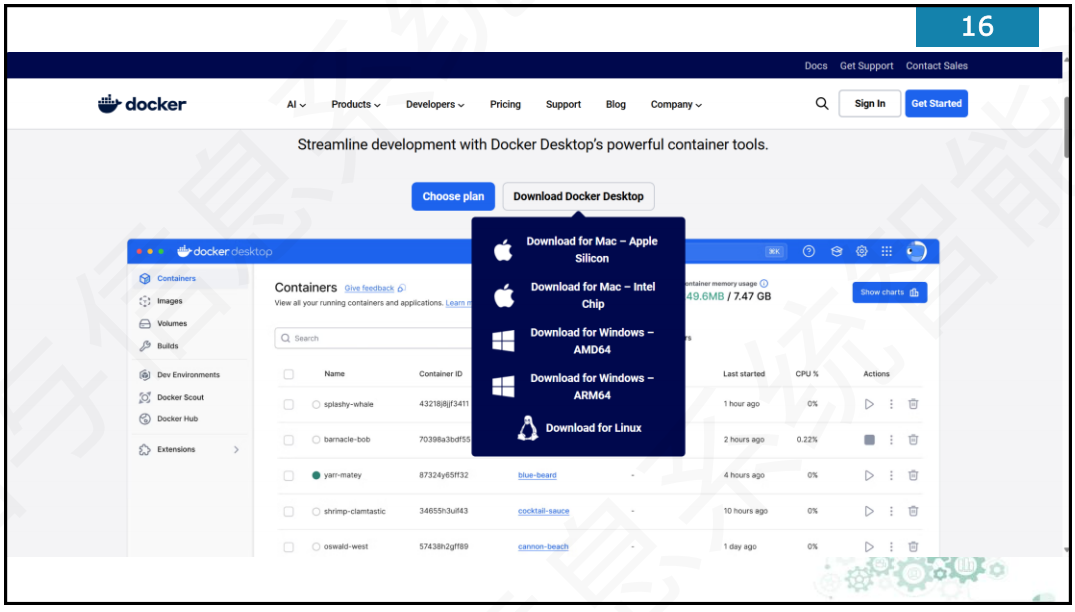
1. 理解容器

- Docker Desktop
  - Docker Desktop是可一键安装的Docker应用程序，支持Mac、Linux或Windows环境，可构建、共享和运行Docker容器。
  - Docker Desktop提供图形界面（GUI），使容器、应用、镜像的管理更加方便。
  - Docker Desktop包括的组件
    - Docker Engine
    - Docker CLI client
    - Docker Scout (additional subscription may apply)
    - Docker Buildx
    - Docker Extensions
    - Docker Compose
    - Docker Content Trust
    - Kubernetes
    - Credential Helper

15


Docs
Get Support
Contact Sales
docker
AI
Products
Developers
Pricing
Support
Blog
Company
Sign In
Get Started
docker desktop
The #1 containerization software for developers and teams
Streamline development with Docker Desktop's powerful container tools.
Choose plan
Download Docker Desktop
Containers
Images
Container CPU usage
0.01% / 1200%
Container memory usage
249.6MB / 7.47 GB
Show charts

16


Docs
Get Support
Contact Sales
docker
AI
Products
Developers
Pricing
Support
Blog
Company
Sign In
Get Started
Streamline development with Docker Desktop's powerful container tools.
Choose plan
Download Docker Desktop
Download for Mac – Apple Silicon
Download for Mac – Intel Chip
Download for Windows – AMD64
Download for Windows – ARM64
Download for Linux
Containers
Images
Volumes
Builds
Dev Environments
Docker Scout
Docker Hub
Extensions
Name
Container ID
Last started
CPU %
Actions
splashy-whale
barnacle-bob
yarn-matey
shrimp-clamtastic
oswald-west
blue-beard
cocktail-sauce
cannon-beach
1 hour ago
2 hours ago
4 hours ago
10 hours ago
1 day ago
Show charts

Docker Desktop enhances your development experience by offering a powerful, user-friendly platform for container management. Fully integrated with your development tools, it simplifies container deployment and accelerates your workflow efficiency.

Docker Engine

Powerful container runtime

The Docker Engine powers your containerized applications with high performance and reliability. It provides the core technology for building and running containers, ensuring efficient and scalable operations.

Docker CLI

Flexible command-line interface

The Docker CLI offers a robust command-line tool for precise control over your containers. Execute complex commands, automate tasks, and integrate Docker seamlessly into your workflows.

Docker Compose

Streamlined multi-container management

Docker Compose simplifies the process of managing multi-container applications. Define and run complex setups with a single configuration file, making it easier to deploy and scale your applications.

Docker Build

Simplified container building

Docker Build is a powerful tool within Docker Desktop that simplifies the process of creating container images. It enables you to package and build your code to ship it anywhere while integrating seamlessly into your development pipeline.

Docker Kubernetes

Built-in container orchestration

Docker Kubernetes provides built-in Kubernetes support within Docker Desktop, allowing you to orchestrate and manage containers efficiently. Supporting both multi-node clusters and developer-selected versions, Docker Kubernetes simplifies deploying, scaling, testing, and

Volume Management

Effective data management

Docker Volumes provides a robust solution for managing and sharing container data. This feature allows you to easily and securely manage volumes for backup, sharing, or migration purposes, enhancing data management and portability.

Synchronized File Shares

Seamless data synchronization

Synchronized File Shares enable real-time sharing and synchronization of files between your host and containers. This feature ensures that file updates are instantly reflect on the host and container, improving collaboration and consistency.

Docker Debug

Advanced troubleshooting tools

Docker Debug provides comprehensive tools for diagnosing and resolving issues within your containers and images. This CLI command lets you create and work with slim containers that would otherwise be difficult to debug.

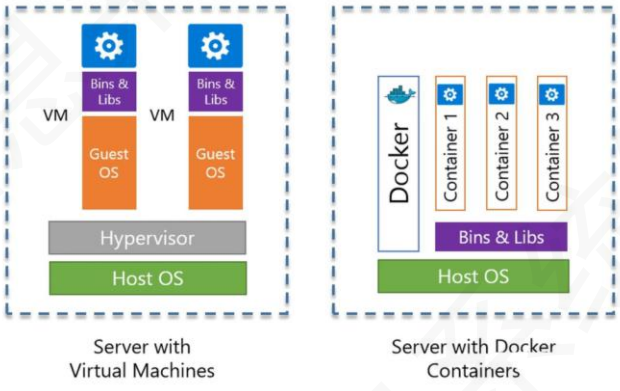
1. 理解容器

- Docker核心概念：镜像(Image)、容器(Container)、仓库(Repository)
  - 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需的文件和依赖包。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。
    - 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，网址是https://hub.docker.com。

1. 理解容器

- Docker容器和虚拟机有很多相似的地方，比如在资源隔离和分配优势方面，但其功能并不相同。
  - Docker容器虚拟的是操作系统，虚拟机虚拟的是硬件。
    - 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。
  - Docker容器没有操作系统，虚拟机有独立的操作系统。
    - 虚拟机是虚拟出一套硬件后，在其上运行一个完整的操作系统，在该系统上再运行所需应用进程。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容器与虚拟机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。如云服务商通常采用虚拟机技术隔离不同用户。
    - Docker技术通常用于隔离不同的应用。如前端、后端以及数据库的部署。

容器与虚拟机对比



容器与虚拟机对比

1. 理解容器

- 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，不会波及整台机器。

1. 理解容器

- Docker的应用场景
  - 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。
  - Docker的主要应用场景：
    - 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

2. 通过Docker实现容器应用

安装Docker

任务目标：

- 在openEuler上安装Docker软件，进行Docker服务的测试与管理。

操作步骤：

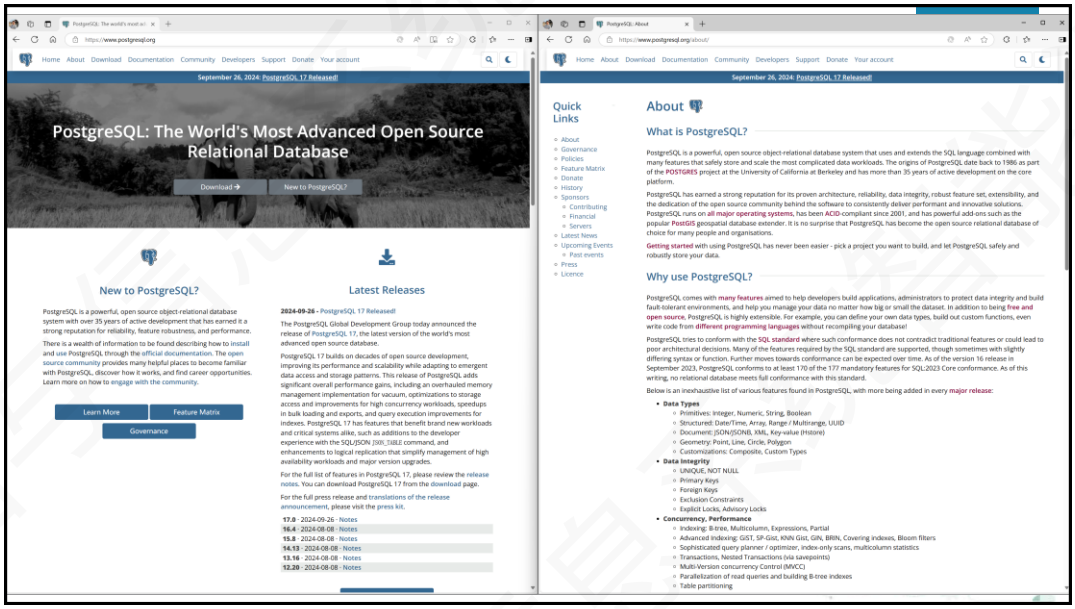
- 通过二进制方式安装Docker软件
- 启动Docker服务
- 查看Docker运行信息
- 验证

操作演示：

Code

2. 通过Docker实现容器应用

- PostgreSQL是一款卓越的开源关系型数据库管理系统。
  - 功能方面
    - 具备丰富的数据类型，不仅有常见的数值、字符串、日期时间类型，还支持数组、JSON、XML、几何图形等复杂数据类型，方便存储和处理多样化的数据。
    - 拥有强大的查询功能，支持复杂的 SQL 查询操作、窗口函数和公共表达式等高级特性。提供多种索引类型和优秀的查询优化器，提升查询性能。
  - 事务处理
    - 严格遵循事务的 ACID 属性，确保数据的完整性和一致性。采用多版本并发控制机制，允许多用户同时访问和修改数据而互不干扰。
    - 支持存储过程和函数，可将复杂业务逻辑封装在数据库中，提高效率和可维护性。
  - 具有良好的扩展性
    - 可通过插件添加新功能。
    - 提供高可用性解决方案和丰富的备份恢复工具。
    - 有强大的安全机制，包括用户认证、访问控制和数据加密等。



2. 通过Docker实现容器应用

基于Docker实现PostgreSQL数据库服务

任务目标：

- 使用Docker实现PostgreSQL数据库服务。

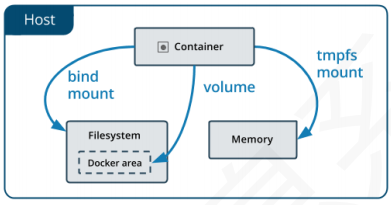
操作步骤：

- 拉取PostgreSQL镜像
- 创建PostgreSQL的数据卷
- 启动PostgreSQL容器
- 管理PostgreSQL容器
- 使用PgAdmin远程连接PostgreSQL数据库

操作演示：

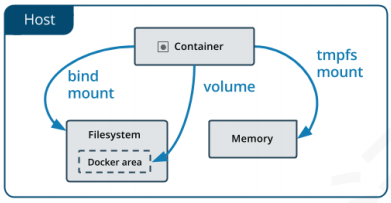
2. 通过Docker实现容器应用

- 默认情况下，容器内创建的所有文件都存储在容器的可写层上。
  - 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。
- Docker支持3种方式的数据存储：数据卷(volume)、目录挂载(bind mount)、内存文件系统挂载(tmpfs)。
  - 数据卷和目录挂载可实现持久化存储。
  - 内存文件系统挂载将随容器停止而删除，无法持久化。



2. 通过Docker实现容器应用

- Docker支持3种方式的数据存储。
  - 数据卷（volume）由Docker管理，是Docker宿主机文件系统中的一部分。
    - 在Linux系统中默认的目录为/var/lib/docker/volumes/，非Docker进程不应修改此文件系统，数据卷是Docker推荐的数据持久化方式。
  - 目录挂载（bind mount）可存储数据至Docker宿主机的任何地方。
    - Docker宿主机或者Docker容器中的非Docker进程可随时修改此文件系统。
  - 内存文件系统挂载（tmpfs）仅存储在Docker宿主机系统的内存中，且永远不会写入Docker宿主机的文件系统。



2. 通过Docker实现容器应用

- cAdvisor（Container Advisor）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旨在监控正在运行的容器的资源使用情况和性能特征。
  - cAdvisor是一个正在运行的守护进程，收集、聚合、处理和导出有关正在运行的容器的信息。具体来说，它为每个容器保存资源隔离参数、历史资源使用情况、完整历史资源使用情况的直方图和网络统计信息。这些数据按容器和机器范围导出。
  - cAdvisor对Docker容器有原生支持，支持开箱即用。

2. 通过Docker实现容器应用

cAdvisor监控Docker容器

任务目标：

- 通过cAdvisor工具实现对Docker容器的监控与性能分析。

操作步骤：

- 安装cAdvisor工具
- 监控数据解读

操作演示：

Code

2. 通过Docker实现容器应用

- 使用cAdvisor可监控Docker宿主机的运行情况与容器的运行情况。
  - Docker宿主机的运行情况监控。
    - 在Docker宿主机运行情况监控中，可监控Docker宿主机CPU、内存、网络、文件的使用情况，Docker宿主机所运行容器的CPU使用率、内存使用率排行。
  - Docker容器列表。
    - 单击“Docker Containers”命令可查看Docker容器的运行情况，其中包含Docker容器列表和Docker环境状态。
  - Docker运行情况监控。
    - 单击Docker容器列表中的某个容器名称可查看容器运行情况，其中包括Docker容器的CPU、内存、网络、文件的使用情况。

表 10-4-1　Docker 运行情况监控

监控对象	监控内容	监控说明
Overview 概览	CPU	CPU 使用率
	Memory	内存使用率
	FS#1	存储空间使用率
CPU CPU 使用情况	Total Usage	CPU 总使用率
	Usage per Core	每个 CPU 核心的使用率
	Usage Breakdown	CPU 使用详情
Memory 内存使用情况	Total Usage	内存使用量
	Usage Breakdown	内存使用详情，包含内存总量和已使用量
Network 网络使用情况	Throughput	网络吞吐量
	Errors	网络包错误数
Filesystem 存储空间使用情况	FS #1 tmpfs	列出所有存储空间，并展示每个存储空间总大小，使用大小与使用率
Subcontainers 容器使用情况	Top CPU Usage	CPU 使用最高的容器，默认展示 10 个

3. 使用Docker Compose部署容器应用

- Docker Compose是一个用于定义和运行多容器应用程序的工具。
  - 可以简化开发和部署过程，并提升效率。
  - Compose简化了整个应用程序堆栈的控制，通过单个YAML配置文件管理服务、网络和存储卷，使用单个命令从配置文件创建和启动所有服务。
  - Compose适用于所有环境，包括生产、开发、测试、迁移、CI工作流，还具有管理应用全生命周期的指令，包括：
    - 启动、停止和重建服务
    - 查看服务的运行状态
    - 流式输出服务的实时日志
    - 一个命令管理服务

学会使用Docker Compose，真用不到三分钟

https://www.bilibili.com/video/BV1ed4y1o7HF

3. 使用Docker Compose部署容器应用

- Drupal是开源的内容管理系统。
  - 由Drupal社区开发和维护。
  - 用PHP编写，可以在Linux、Windows和macOS等操作系统上运行。
  - 功能强大的CMS，拥有丰富的功能和工具，如内容管理、用户管理、权限管理、多语言支持、搜索引擎优化、电子商务支持等。
  - 支持第三方集成，如社交媒体、电子邮件服务等。
  - Drupal的学习曲线可能较陡峭。

3. 使用Docker Compose部署容器应用

使用Docker Compose部署Drupal

任务目标：

- 使用Docker Compose统一管理MySQL和Drupal两个容器，实现内容网站的发布。

操作步骤：

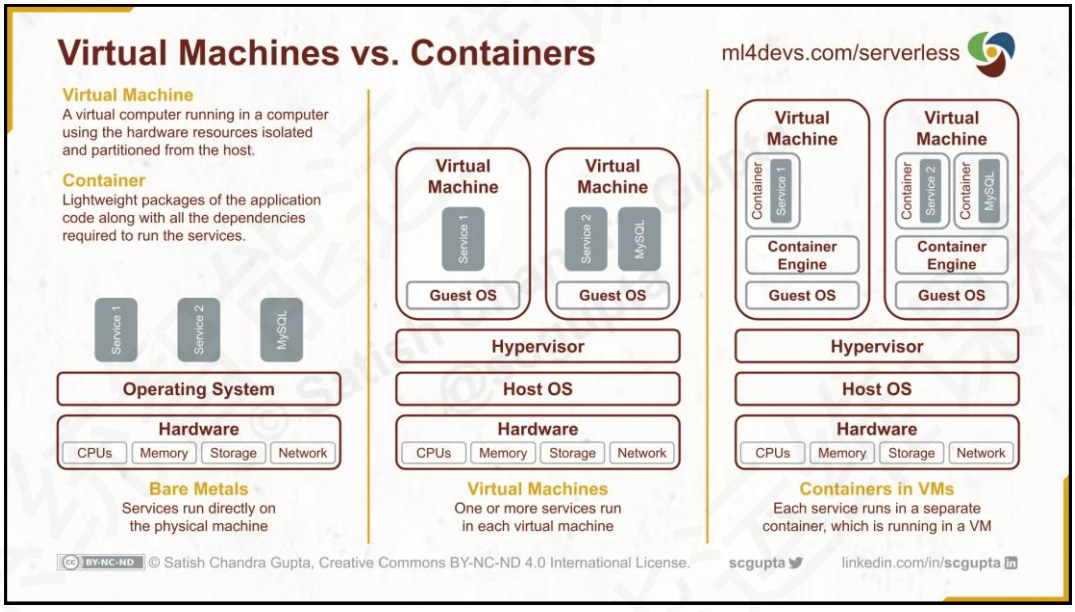
- 安装Docker Compose
- 创建Docker Compose的配置文件
- 使用Docker Compose发布Drupal命令
- 初始化安装Drupal
- 查看容器的运行状态

操作演示：

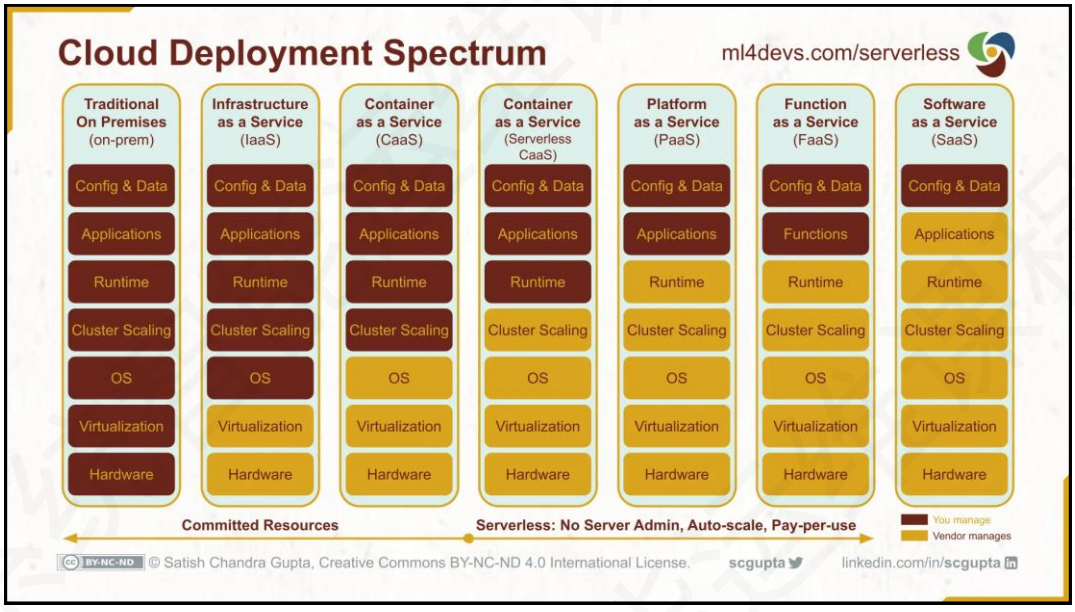
4. 深入理解物理机、虚拟机和容器

选择哪种方式部署业务？

- **裸金属** Bare Metals
- **虚拟化** Virtual Machines
- **容器** Containers


Virtual Machines vs. Containers
ml4devs.com/serverless
Virtual Machine
A virtual computer running in a computer using the hardware resources isolated and partitioned from the host.
Container
Lightweight packages of the application code along with all the dependencies required to run the services.
Service 1
Service 2
MySQL
Operating System
Hardware
CPUs
Memory
Storage
Network
Bare Metals
Services run directly on the physical machine
Virtual Machine
Guest OS
Hypervisor
Host OS
Virtual Machines
One or more services run in each virtual machine
Container
Container Engine
Containers in VMs
Each service runs in a separate container, which is running in a VM
BY-NC-ND
© Satish Chandra Gupta, Creative Commons BY-NC-ND 4.0 International License.
scgupta
linkedin.com/in/scgupta


40
物理机、虚拟机和容器有什么区别?
https://www.bilibili.com/video/BV1nd4y1K7jE


Cloud Deployment Spectrum
ml4devs.com/serverless
Traditional On Premises (on-prem)
Infrastructure as a Service (IaaS)
Container as a Service (CaaS)
Container as a Service (Serverless CaaS)
Platform as a Service (PaaS)
Function as a Service (FaaS)
Software as a Service (SaaS)
Config & Data
Applications
Functions
Runtime
Cluster Scaling
OS
Virtualization
Hardware
Committed Resources
Serverless: No Server Admin, Auto-scale, Pay-per-use
You manage
Vendor manages
BY-NC-ND © Satish Chandra Gupta, Creative Commons BY-NC-ND 4.0 International License.
scgupta
linkedin.com/in/scgupta


42
网络与信息系统智能运维
课程体系学习平台
***
本课程体系由
河南中医药大学信息技术学院建设
课程体系学习平台由河南中医药大学医疗健康信息工程技术研究所开发与技术保障
网络与信息系统智能运维课程体系学习平台
https://internet.hactcm.edu.cn
互联网运维管理工程应用丛书
http://www.51xueweb.cn
扫码学习
并获取课程资源
棒棒堂智能运维
抖音
@棒棒堂智能运维