Overview

1. What is a Container and Why?
2. How Docker helps using containers
3. Docker Commands
4. Orchestration: Swarms and Kubernetes
5. Docker Networking and Security

Advantages of Virtualization

- Minimize hardware costs (CapEx)
  - Multiple virtual servers on one physical hardware
- Easily move VMs to other data centers
  - Provide disaster recovery. Hardware maintenance.
  - Follow the sun (active users) or follow the moon (cheap power)
- Consolidate idle workloads. Usage is bursty and asynchronous.
  - Increase device utilization
- Conserve power
  - Free up unused physical resources
- Easier automation (Lower OpEx)
  - Simplified provisioning/administration of hardware and software
- Scalability and Flexibility: Multiple operating systems

Problems of Virtualization

- Each VM requires an operating system (OS)
  - Each OS requires a license ⇒ CapEx
  - Each OS has its own compute and storage overhead
  - Needs maintenance, updates ⇒ OpEx
  - VM Tax = added CapEx + OpEx
Solution: Containers

- Run many apps in the same virtual machine
  - These apps share the OS and its overhead
  - But these apps can’t interfere with each other
  - Can’t access each other’s resources without explicit permission
  - Like apartments in a complex
  ⇒ Containers

Containers

- Multiple containers run on one operating system on a virtual/physical machine
- All containers share the operating system ⇒ CapEx and OpEx
- Containers are isolated ⇒ cannot interfere with each other
  - Own file system/data, own networking ⇒ Portable

Containers (Cont)

- Containers have all the good properties of VMs
  - Come complete with all files and data that you need to run
  - Multiple copies can be run on the same machine or different machine ⇒ Scalable
  - Same image can run on a personal machine, in a data center or in a cloud
  - Operating system resources can be restricted or unrestricted as designed at container build time
  - Isolation: For example, “Show Process” (ps on Linux) command in a container will show only the processes in the container
  - Can be stopped. Saved and moved to another machine or for later run

VM vs. Containers

<table>
<thead>
<tr>
<th>Criteria</th>
<th>VM</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Size</td>
<td>3X</td>
<td>X</td>
</tr>
<tr>
<td>Boot Time</td>
<td>&gt;10s</td>
<td>~1s</td>
</tr>
<tr>
<td>Computer Overhead</td>
<td>&gt;10%</td>
<td>&lt;5%</td>
</tr>
<tr>
<td>Disk I/O Overhead</td>
<td>&gt;50%</td>
<td>Negligible</td>
</tr>
<tr>
<td>Isolation</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Low-Medium</td>
<td>Medium-High</td>
</tr>
<tr>
<td>OS Flexibility</td>
<td>Excellent</td>
<td>Poor</td>
</tr>
<tr>
<td>Management</td>
<td>Excellent</td>
<td>Evolving</td>
</tr>
<tr>
<td>Impact on Legacy application</td>
<td>Low-Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

Docker

- Provides the isolation among containers
- Helps them share the OS
- Docker = Dock worker \(\Rightarrow\) Manage containers
- Developed initially by Docker.com
- Downloadable for Linux, Windows, and Mac from Docker.com
- Customizable with replacement modules from others

![Docker Diagram](image)

Docker Engine Components

- daemon: API and other features
- containerd: Execution logic. Responsible for container lifecycle. Start, stop, pause, unpause, delete containers.
- runc: A lightweight runtime CLI
- shim: runc exists after creating the container.
  - shim keeps the container running. Keep stdin/stdout open.

Docker Engine

- Two Editions:
  - Community Edition (CE): Free for experimentation
  - Enterprise Edition (EE): For deployment with paid support
- Written in “Go” programming language from Google
- Now open source project under mobyproject.org
  - [https://github.com/moby/moby](https://github.com/moby/moby)
- Download the community edition and explore

Image Registries

- Containers are built from images and can be saves as images
- Images are stored in registries
  - Local registry on the same host
  - Docker Hub Registry: Globally shared
  - Private registry on Docker.com
- Any component not found in the local registry is downloaded from specified location
- Official Docker Registry: Images vetted by Docker
- Unofficial Registry: Images not vetted (Use with care)
- Each image has several tags, e.g., v2, latest, …
- Each image is identified by its 256-bit hash

Docker

- Docker Engine: Runtime
- Community Edition (CE): Free for experimentation
- Enterprise Edition (EE): For deployment with paid support
- Written in “Go” programming language from Google
- Now open source project under mobyproject.org
  - [https://github.com/moby/moby](https://github.com/moby/moby)
- Download the community edition and explore

Ref: [https://golang.org/](https://golang.org/)

Reference:
### Layers

- Each image has many layers
- Image is built layer by layer
- Layers in an image can be inspected by Docker commands
- Each layer has its own 256-bit hash
- For example:
  - Ubuntu OS is installed, then
  - Python package is installed, then
  - a security patch to the Python is installed
- Layers can be shared among many containers

![Image Layers Diagram]

### Building Container Images

- Create a **Dockerfile** that describes the application, its dependencies, and how to run it
  - FROM Alpine
  - LABEL maintainer="xx@gmail.com"
  - RUN apk add –update nodejs nodejs –npm
  - COPY . /src
  - WORKDIR /src
  - RUN nmp install
  - EXPOSE 8080
  - ENTRYPOINT ["node", ".app.js"]

Start with Alpine Linux
Who wrote this container
Use apk package to install nodejs
Copy the app files from build context
Set working directory
Install application dependencies
Open TCP Port 8080
Main application to run

RUN nmp install
Copy . /src
RUN apk add ...
FROM Alpine

Note: WORKDIR, EXPOSE, ENTRYPOINT result in tags. Others in Layers.

### Docker Commands

- **docker container run**: Run the specified image
- **docker container ls**: List running containers
- **docker container exec**: Run a new process inside a container
- **docker container stop**: Stop a container
- **docker container start**: Start a stopped container
- **docker container rm**: Delete a container
- **docker container inspect**: Show information about a container

### Open Container Initiative (OCI)

- A company called CoreOS defined alternative image format and container runtime API’s
- Led to formation of OCI under Linux Foundation to govern container standards
  - OCI Image spec
  - OCI Runtime spec
- Everyone including Docker is now moving to OCI

Ref: [https://www.opencontainers.org/](https://www.opencontainers.org/)
Swarm

- Orchestrating thousands of containers
- Swarm: A group of nodes collaborating over a network
- Two modes for Docker hosts:
  - Single Engine Mode: Not participating in a swarm
  - Swarm Mode: Participating in a Swarm
- A service may run on a swarm
- Each swarm has a few managers that dispatch tasks to workers. Managers are also workers (i.e., execute tasks)

Swarm

Single-Engine Node

Swarm Node

Swarm Node

Swarm Node

Swarm Node

Swarms (Cont)

- The managers select a leader, who really keeps track of the swarm
- Assigns tasks, re-assigns failed worker’s tasks, …
- Other managers just monitor passively and re-elect a leader if leader fails
- Services can be scaled up or down as needed

- Several Docker commands:
  - `docker service`: Manage services
  - `docker swarm`: Manage swarms
  - `docker node`: Manage nodes

Docker Swarm Commands

- `docker swarm init`
- `docker swarm join-token`
- `docker node ls`
- `docker service create`
- `docker service ls`
- `docker service ps`
- `docker service inspect`
- `docker service scale`
- `docker service update`
- `docker service rm`

Docker Overlay Networking

- Nodes in a swarm may not be in the same LAN
- VXLAN is used to provide virtual overlay networking
- VXLAN was discussed in another module of this course

172.116.56.67 172.118.56.67 192.168.0.1 192.168.0.2

Node 1  Node 2  Node 1  Node 2

Physical Virtual
Docker Security

- All built-in security mechanisms in Linux are used and more
- Cryptographic node IDs
- Mutual Authentication
- Automatic Certificate Authority configuration
- Automatic Certificate Renewal on expiration
- Encrypted Cluster Store
- Encrypted Network traffic
- Signed images in Docker Content Trust (DCT)
- Docker Security Scanning detects vulnerabilities
- Docker secrets are stored in encrypted cluster store, encrypted transmission over network, and stored in in-memory file system when in use

Kubernetes

- Open Source Container Orchestration alternative
- Original source released by Google
- Cloud Native Computing Foundation (CNCF) project in Linux Foundation
- Pre-cursor to Swarms
- Facilities similar to Swarms
- A set of related containers is called a “Pod”
  - A Pod runs on a single host.
- Swarm is called a “Cluster”

Hyper-V Containers

- Microsoft allows two kinds of containers:
  - Windows Server Containers: Multiple containers on a single VM (like Docker containers)
  - Hyper-V containers: Each container runs on its own VM
    => No need for a Linux

<table>
<thead>
<tr>
<th>VM</th>
<th>VM</th>
<th>VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper-V</td>
<td>Container</td>
<td>Container</td>
</tr>
<tr>
<td>Hardware</td>
<td>Hardware</td>
<td>Hardware</td>
</tr>
</tbody>
</table>

Intel Clear Containers

- Started 2015 to address security concerns (Dirty COW) in containers
- Idea: Allow lightweight VMs using Intel Virtualization Technology
  - Own lightweight OS and a dedicated kernel
    => Isolation of network, memory, and I/O
  - Help by hardware enforced isolation
  - No need for full VMs for containers
- Merged with HyperV to form Kata containers on Dec 5, 2017
Kata Containers

- Lightweight virtual machines
- Dedicated VMs to run one and only one container
- Combines “Intel Clear Containers” and “HyperV runV”
- Open source project under OpenStack Foundation
- Compatible with the OCI specs for Docker containers
- Compatible with CRI for Kubernetes
- Performance like containers, isolation and security like VMs
- Six Components: Agent, Runtime, Proxy, Shim, Kernel and QEMU 2.9
- Kubernetes will be extended to provision VMs (Kata Containers)
- OpenStack’s VM orchestration engine (Nova) will be extended to handle containers
- Package once and run anywhere
  - VMware, Google, and Amazon are all moving towards this approach
- No installable distribution of Kata containers yet (April 22, 2018)

Ref: https://katacontainers.io/
Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-18/ ©2018 Raj Jain

Summary

- Virtual Machines provide scalability, mobility, and cost reduction but need OS which increase resource requirements
- Containers provide isolation on a single OS and are lightweight
- Docker allows managing containers
- Docker Swarm and Kubernetes allow orchestrating a large number of containers
- Docker provides overlay networking and security

Acronyms

- API Application Programming Interface
- CapEx Capital Expenditure
- CE Community Edition
- CLI Command Line Interface
- CNCF Native Computing Foundation
- DCT Docker Content Trust
- EE Enterprise Edition
- ID Identifier
- ISBN International Standard Book Number
- LAN Local Area Network
- OpEx Operational Expenses
- OS Operating System
- TCP Transmission Control Protocol
- VM Virtual Machine

References

Wikipedia Links

- https://en.wikipedia.org/wiki/Microservices

Scan This to Download These Slides

Thank you

Raj Jain
http://rajjain.com

Related Modules

CSE567M: Computer Systems Analysis (Spring 2013),
https://www.youtube.com/playlist?list=PLjGG94etKypJWEkNyN9hSs_HCd5e4wXF

CSE473S: Introduction to Computer Networks (Fall 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypJWQSMh8Azegy5e_10TiDw

Wireless and Mobile Networking (Spring 2016),
https://www.youtube.com/playlist?list=PLjGG94etKypKvzfVtuHcPFXumyyg93u

CSE571S: Network Security (Fall 2011),
https://www.youtube.com/playlist?list=PLjGG94etKypKvzfVtuHcPFXumyyg93u

Video Podcasts of Prof. Raj Jain's Lectures,
https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw