

Containers and Virtualization Technologies

Hideo Suzumiya (Gaoyang Wei)

College of Information Engineering (College of Artificial Intelligence)

March 2, 2023

Outline

- 1 Introduction
- 2 Background
- 3 Virtualization
- 4 Containers
- 5 Comparison
- 6 Docker Practice
- 7 Conclusion

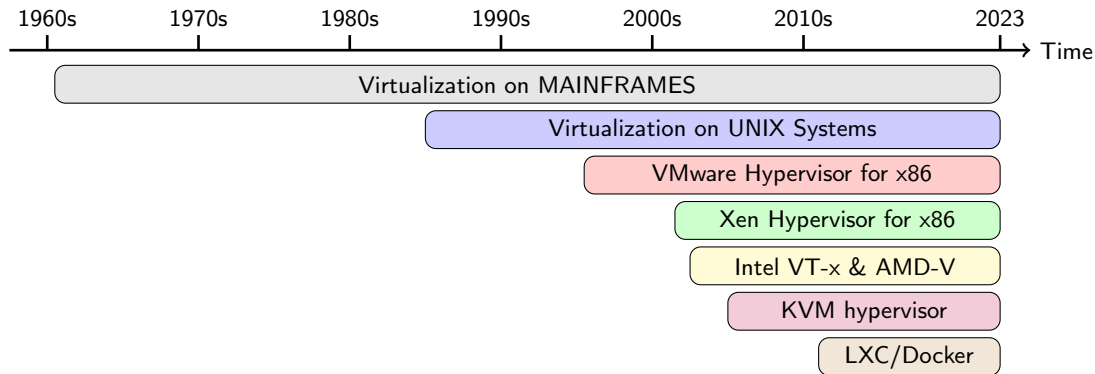
What is Virtualization ?

Virtualization is a technology that allows multiple virtual instances of hardware, operating systems, and software applications to run on a single physical server, creating a virtual environment that appears to be a separate and independent computer system.

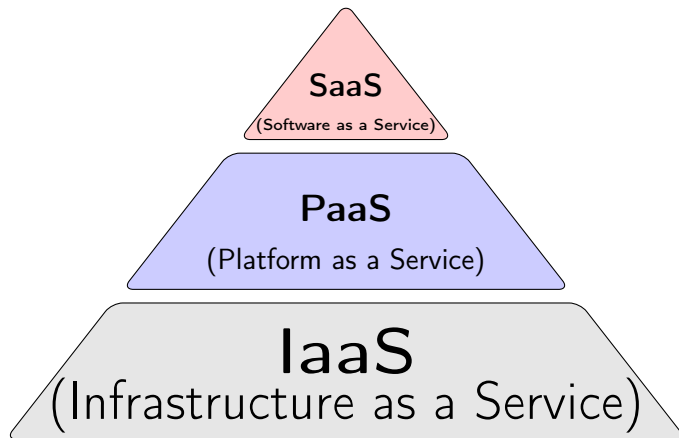
5 Levels of Virtualization Technologies

- Instruction Set Architecture Level *e.g.* QEMU
- Hardware Abstraction Level *e.g.* VMWare ESXi, Microsoft Hyper-V, KVM
- Operating System Level *e.g.* LXC, Docker
- Programming Language Level *e.g.* JVM, Microsoft dotNet
- Library Level *e.g.* Wine, WSL1

Evolution of Virtualization

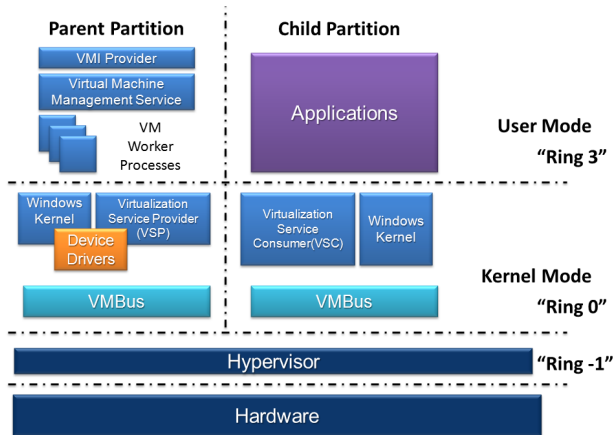
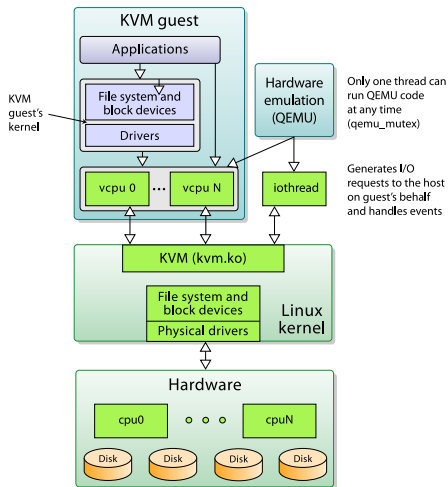


Cloud Computing Models



- Virtualization enables multiple virtual machines (VMs) to run on a single physical server, each with its own operating system and applications.
- Virtualization provides a way to consolidate servers, reduce hardware costs, and increase resource utilization.
- Popular virtualization platforms include VMware, Hyper-V, and Kernel-based Virtual Machine.

KVM & Hyper-V Architectures



Virtualization

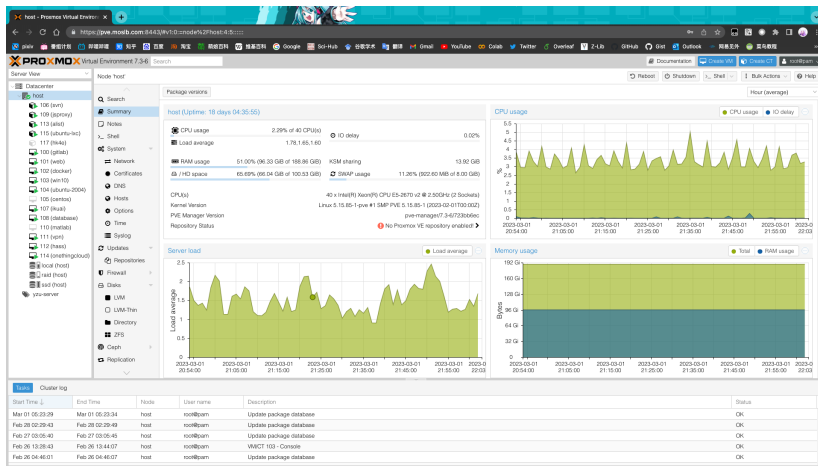


Figure: Proxmox VE 7.3 based on KVM and LXC

What is Container?

- Containers provide a way to package and deploy applications in a lightweight, portable, and reproducible way.
- Containers are Operating System Level VMs!
- Docker is a popular containerization platform that enables developers to build, ship, and run distributed applications in containers.

Philosophy of Docker

- Write once, run everywhere.
- One container runs one application.

Containers

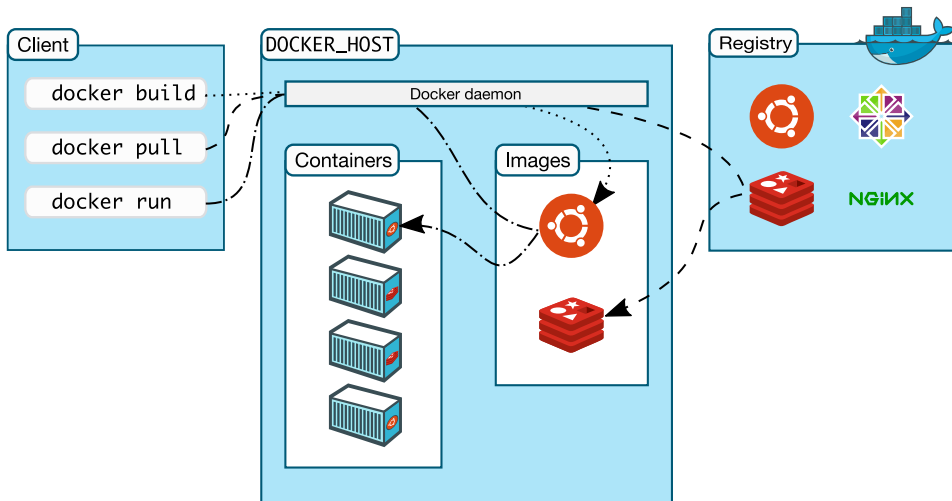
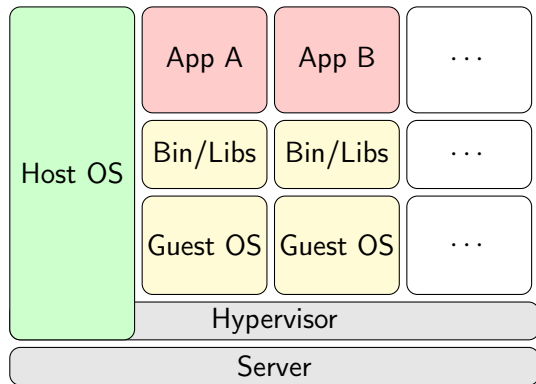
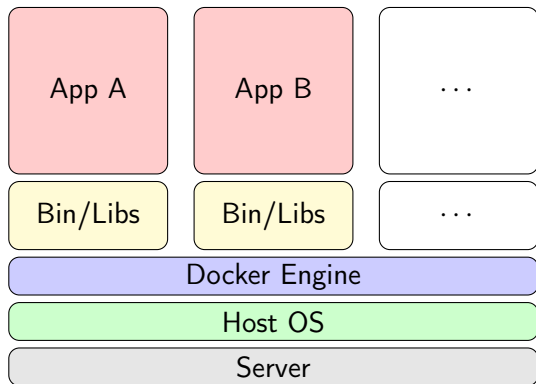


Figure: Docker Architecture

Containers V.S. VMs



Virtual Machines



Containers

Comparison

- Containers are faster, more lightweight, and more efficient than VMs.
- VMs provide greater isolation and security than containers.
- Containers are better suited for microservices-based architectures, while VMs are better suited for legacy applications and monolithic architectures.

Attribute	Container	VM
Start	Seconds	Minutes
Size	10s of Megabytes	10s of Gigabytes
Performance	Close to native	Weaker
Supported Quantity	1000s of containers	10s of Guests

Table: The comparison

Steps of installing Docker on linux

Running Docker on Windows & macOS is **NOT** recommended!

- Change source repo
`export DOWNLOAD_URL="https://mirrors.bfsu.edu.cn/docker-ce"`
- Use auto installation script
`curl -fsSL https://get.docker.com/ | sh`
- Or
`wget -O- https://get.docker.com/ | sh`
- Change DockerHub repo
`touch /etc/docker/daemon.json`

```
{  
  "registry-mirrors": [  
    "https://hub-mirror.c.163.com",  
    "https://mirror.baidubce.com" ]  
}
```

Install Docker

- Reload systemd services

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```
- Execute `docker info` and if you see

Registry Mirrors:

<https://hub-mirror.c.163.com/>

The configuration has been applied.

Command	Description
<code>docker create <i>image</i> [<i>command</i>]</code>	Create the container
<code>docker run <i>image</i> [<i>command</i>]</code>	= create + start
<code>docker start <i>container</i> ...</code>	Start the container
<code>docker stop <i>container</i> ...</code>	Graceful stop
<code>docker kill <i>container</i> ...</code>	Kill (SIGKILL) the container
<code>docker restart <i>container</i> ...</code>	= stop + start
<code>docker pause <i>container</i> ...</code>	Suspend the container
<code>docker unpause <i>container</i> ...</code>	Resume the container
<code>docker rm [-f] <i>container</i> ...</code>	Destroy the container

Command	Description
<code>docker ps</code>	List running containers
<code>docker ps -a</code>	List all containers
<code>docker logs [-f] <i>container</i> ...</code>	Show the container output (stdout & stderr)
<code>docker top <i>container</i> [<i>ps options</i>]</code>	List the process running inside the container
<code>docker diff <i>container</i></code>	Show the difference with images (modified files)
<code>docker inspect <i>container</i> ...</code>	Show low-level infos (in json format)

Command	Description
<code>docker attach <i>container</i></code>	attach to a running container (stdin/stdout/stderr)
<code>docker cp <i>container:path hostpath</i> -</code>	Copy files from container
<code>docker cp <i>hostpath</i> - <i>container:path</i></code>	Copy files to container
<code>docker export <i>container</i></code>	export the content of the container (Tar archive)
<code>docker exec <i>container</i> ...</code>	Run a command in an existing container
<code>docker wait <i>container</i></code>	Wait until the container terminates and return the exit code
<code>docker commit <i>container image</i></code>	Commit a new docker image (snapshot or container)

Command	Description
<code>docker images</code>	List all local images
<code>docker history <i>image</i></code>	Show image history (list of ancestors)
<code>docker inspect <i>image</i> ...</code>	Show low-level infos (in json format)
<code>docker tag <i>image tag</i></code>	Tag an image
<code>docker commit <i>container image</i></code>	Create image from container
<code>docker import <i>url</i> - [<i>tag</i>]</code>	Create image from tarball
<code>docker rmi <i>image</i> ...</code>	Delete images

Command	Description
<code>docker pull <i>repo[:tag]</i> ...</code>	Pull an image/repo from a registry
<code>docker push <i>repo[:tag]</i> ...</code>	Push an image/repo from a registry
<code>docker search <i>text</i></code>	Search an image from the official registry
<code>docker login ...</code>	Login to a registry
<code>docker logout ...</code>	Logout from a registry

[Table:](#) Using the registry API

Command	Description
<code>docker save <i>repo[:tag]</i> ...</code>	Export an image/repo as a tarball
<code>docker load</code>	Load image from a tarball
<code>docker-ssh ...</code>	Proposed script to transfer images between 2 daemons over ssh

Table: Manual transfer

Command	Description
FROM <i>image scratch</i>	Base image for the build
MAINTAINER <i>email</i>	Name of maintainer (metadata)
COPY <i>path dst</i>	Copy <i>path</i> from the context into the container at location <i>dst</i>
ADD <i>src dst</i>	Same as COPY but untar archives and accepts http urls
RUN <i>args ...</i>	Run an arbitrary command inside the container
USER <i>name</i>	Set the default username
WORKDIR <i>path</i>	Set the default working directory
CMD <i>args ...</i>	Set the default command
ENV <i>name value</i>	Set environment variables

- Containerization and virtualization are both important technologies for modern application development and deployment.
- Containers and VMs have different strengths and weaknesses, and choosing the right technology depends on the specific requirements of the application and infrastructure.

Conclusion


 Cheatsheet for Docker CLI			
Run a new Container	Manage Containers	Manage Images	Info & Stats
<p>Start a new Container from an Image</p> <pre>docker run IMAGE docker run nginx</pre> <p>...and assign it a name</p> <pre>docker run --name CONTAINER IMAGE docker run --name web nginx</pre> <p>...and map a port</p> <pre>docker run -p HOSTPORT:CONTAINERPORT IMAGE docker run -p 8080:80 nginx</pre> <p>...and map all ports</p> <pre>docker run -P IMAGE docker run -P nginx</pre> <p>...and start container in background</p> <pre>docker run -d IMAGE docker run -d nginx</pre> <p>...and assign it a hostname</p> <pre>docker run --hostname HOSTNAME IMAGE docker run --hostname srv nginx</pre> <p>...and add a dns entry</p> <pre>docker run --add-host HOSTNAME:IP IMAGE</pre> <p>...and map a local directory into the container</p> <pre>docker run -v HOSTDIR:TARGETDIR IMAGE docker run -v ~/.usr/share/nginx/html nginx</pre> <p>...but change the entrypoint</p> <pre>docker run -it --entrypoint EXECUTABLE IMAGE docker run -it --entrypoint bash nginx</pre>	<p>Show a list of running containers</p> <pre>docker ps</pre> <p>Show a list of all containers</p> <pre>docker ps -a</pre> <p>Delete a container</p> <pre>docker rm CONTAINER docker rm web</pre> <p>Delete a running container</p> <pre>docker rm -f CONTAINER docker rm -f web</pre> <p>Delete stopped containers</p> <pre>docker container prune</pre> <p>Stop a running container</p> <pre>docker stop CONTAINER docker stop web</pre> <p>Start a stopped container</p> <pre>docker start CONTAINER docker start web</pre> <p>Copy a file from a container to the host</p> <pre>docker cp CONTAINER:SOURCE TARGET docker cp web:/index.html index.html</pre> <p>Copy a file from the host to a container</p> <pre>docker cp TARGET CONTAINER:SOURCE docker cp index.html web:/index.html</pre> <p>Start a shell inside a running container</p> <pre>docker exec -it CONTAINER EXECUTABLE docker exec -it web bash</pre> <p>Rename a container</p> <pre>docker rename OLD_NAME NEW_NAME docker rename 090 web</pre> <p>Create an image out of container</p> <pre>docker commit CONTAINER docker commit web</pre>	<p>Download an image</p> <pre>docker pull IMAGE[:TAG] docker pull nginx</pre> <p>Upload an image to a repository</p> <pre>docker push IMAGE docker push myimage:1.0</pre> <p>Delete an image</p> <pre>docker rmi IMAGE</pre> <p>Show a list of all images</p> <pre>docker images</pre> <p>Delete dangling images</p> <pre>docker image prune</pre> <p>Delete all unused images</p> <pre>docker image prune -a</pre> <p>Build an image from a Dockerfile</p> <pre>docker build DIRECTORY docker build .</pre> <p>Tag an image</p> <pre>docker tag IMAGE NEWIMAGE docker tag ubuntu ubuntu:18.04</pre> <p>Build and tag an image from a Dockerfile</p> <pre>docker build -t IMAGE DIRECTORY docker build -t myimage .</pre> <p>Save an image to a tar file</p> <pre>docker save IMAGE > FILE docker save nginx > nginx.tar</pre> <p>Load an image from a .tar file</p> <pre>docker load -i TARFILE docker load -i nginx.tar</pre>	<p>Show the logs of a container</p> <pre>docker logs CONTAINER docker logs web</pre> <p>Show stats of running containers</p> <pre>docker stats</pre> <p>Show processes of container</p> <pre>docker top CONTAINER docker top web</pre> <p>Show installed docker version</p> <pre>docker version</pre> <p>Get detailed info about an object</p> <pre>docker inspect NAME docker inspect nginx</pre> <p>Show all modified files in container</p> <pre>docker diff CONTAINER docker diff web</pre> <p>Show mapped ports of a container</p> <pre>docker port CONTAINER docker port web</pre>

Figure: Docker Cheatsheet:

<https://raw.githubusercontent.com/sangam14/dockercheatsheets/master/dockercheatsheet8.png>

Thanks!