Docker Introduction *

Jens Lechtenbörger

VM Neuland im Internet 2021

Contents

1 Introduction

1.1 Motivation (1/2)

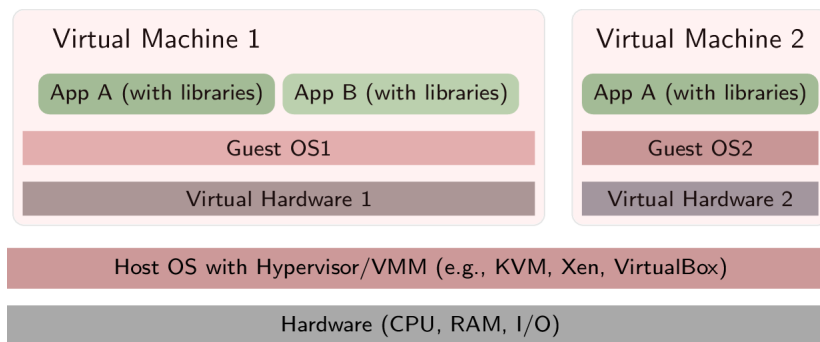- Virtualization software provides **virtual hardware**

Figure 1: Layering with virtualization

- Virtualization implemented by piece of software called Hypervisor/VMM
    - VMM runs on (usual) host OS, translates access to real hardware
- Virtual hardware can have **arbitrary** features

---

*This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

* Largely independent of real hardware, say, ten network cards
- On top of virtual hardware, install operating systems (guests) and other software to create virtual machines (VMs)
  * **Share resources** of powerful server machine among several VMs
    · E.g., your "own" server as VM in a project seminar
  * Use VM as **blueprint** to share reliable environment with others
    · Or to fire up lots of **identical** VMs for compute-intensive tasks with cloud computing

This and the subsequent slide are intended as quick overview for virtualization and containerization. Terms used here as well as the layered figure are revisited later on.

1.2 Motivation (2/2)

- Containerization (e.g., with Docker) as lightweight variant of virtualization

Container 1	Container 2	Container 3
App A (with libraries)	App B (with libraries)	App A (with libraries)
Environment 1 based on kernel of Host OS	Environment 2 based on kernel of Host OS	Environment 1 based on kernel of Host OS

Container manager (e.g., Docker)

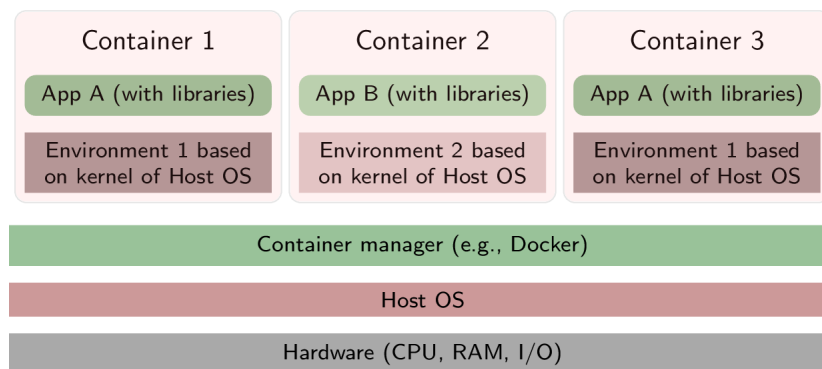| Host OS |
|---|

Hardware (CPU, RAM, I/O)

Figure 2: Layering with containerization

- No virtual hardware, but shared OS kernel
- Use containers to execute software (versions) in controlled way
  * Think of larger application that uses external libraries
  * Libraries evolve, may introduce incompatible changes over time
    · Specific version of application depends on specific versions of libraries
    · Container bundles "correct" versions

1.3 Learning Objectives

- Explain definitions of virtual machine and virtual machine monitor

- Explain and contrast virtualization and containerization

  - Including isolation
  - Including layering

- Use Docker for simple tasks

  - E.g., start Web/Solid server with static files
  - Interpret and modify simple docker files

1.4 Core Questions

- What do virtualization and containerization mean?

- How to deploy potentially complex software in a reproducible fashion?

2 Virtualization

2.1 History (1/2)

- Virtualization is an old concept

  - IBM mainframes, 1960s

  - Frequently cited survey article by Goldberg, 1974: [Gol74]

  - Original motivation

    * Resources of **expensive** mainframes better utilized with multiple VMs

    * Ability to run different OS versions in parallel, **backwards compatibility**

- 1980s, 1990s

  - Modern multitasking OSs on **cheap** hardware

    * Cheap hardware did not offer virtualization support

    * Little use of virtualization

2.2 History (2/2)

- Ca. 2005

  - PC success becomes **problematic**

    * How to limit **energy usage** and **management overhead** of fleets of PCs in data centers?

  - One answer: Use virtualization for **server consolidation**

    * Turn independent servers into VMs, then allocate them to single server

      · Servers often with low resource utilization (e.g., CPU usage between 10% and 50% at Google in 2007, [BH07])

      · Consolidated server with improved resource utilization

  - Additional answer: Virtualization reduces management, testing, and deployment overhead, see [Vog08] for Amazon

  - Virtualization as enabler for cloud computing

- [Sol+07]: Containers for lightweight virtualization

- [Cas+19]: Serverless computing (beyond our scope)

2.3 Intuition and Examples

- Virtualization: Creation of virtual/abstract version of something

  - Virtual memory, recall OS concepts
    * Not our focus
  - Network, e.g., overlay networks, software-defined networking
    * Not our focus
  - Execution environment (e.g., Java, Dotnet)
  - Hardware/system: virtual machine (VM)

- Typical meaning: **virtual machine** (VM)

  - Virtual hardware
    * Several OSs share same underlying hardware
  - VMs isolated from each other

2.4 Definitions

- Cited from [PG74] (bold face added)

  - "A **virtual machine** is taken to be an *efficient, isolated duplicate* of the real machine."
  - Made precise with **Virtual Machine Monitor** (VMM)
    * "First, the VMM provides an **environment** for programs which is **essentially identical** with the original machine; second, programs run in this environment show at worst only **minor decreases in speed**; and last, the VMM is in **complete control** of system resources."
      · Essentially identical: Programs with same results, maybe different timing
      · Speed: Most instructions executed directly by CPU with no VMM intervention
      · Control: (1) Virtualized programs restricted to resources allocated by VMM, (2) VMM can regain control over allocated resources
    * "A *virtual machine* is the environment created by the virtual machine monitor."

2.5 Isolation

- Isolation of VMs: Illusion of exclusive hardware use (despite sharing between VMs)

  - Related to "isolated duplicate" and "complete control" of [PG74]

- Sub-types (see [Sol+07; Fel+15])

  - Resource isolation: Fair allocation and scheduling

* Reservation (e.g., number of CPU cores and amount of RAM) vs best-effort
- Fault isolation: Buggy component should not affect others
- Security isolation
  * Configuration independence (global names/settings do not conflict)
    · Applications with conflicting requirements for system-wide configuration
    · E.g., port 80 for Web servers, each application with own version of shared libraries
  * Safety (no access between VMs/containers)
  * Beware! Lots of security issues in practice
    · E.g., hypervisor privilege escalation and cross-VM side channel attacks
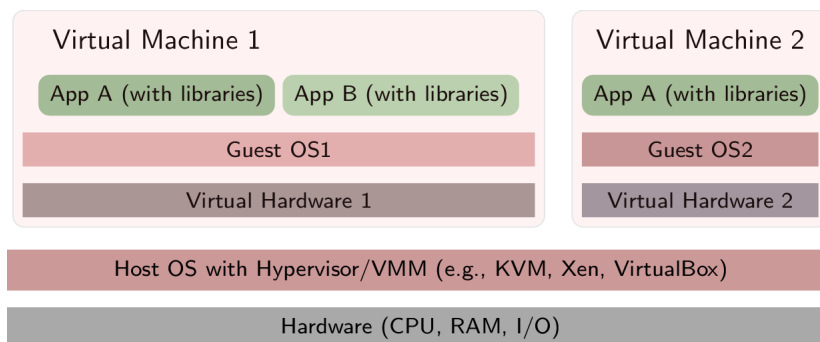
2.6 Layering with Virtualization

Figure 3: Layering with virtualization

2.6.1 Layering Explained

- Hypervisor or virtual machine manager (VMM) with full access to physical hardware
  - Most privileged code
    * Details depend on CPU hardware
      · E.g., kernel mode (CPU ring 0) or additional "root mode" with more privileges than kernel mode
  - Create abstract versions of hardware, to be used by **guest OSs**
    * VM = Guest OS running on abstract hardware
    * Host = Environment in which the VMM runs
      · Host software may be full OS or specialized

- Guest OS is **de-privileged**
  - No longer with full hardware access, e.g., CPU ring 1

- – Privileged/sensitive instructions lead to hypervisor
    - ∗ Executed, translated, or emulated accordingly

- Each VM can run different OS

- VM backups/snaphots **simplify** management, placement, parallelization

- Sharing among applications in different VMs **restricted**, requires networking

    - – (Neither shared memory nor file nor pipes)

- Creation of more VMs with **high overhead**

    - – Each with full OS, own portion of underlying hardware

2.7 Review Question

- The Java VM was mentioned as variant of virtualization. Discuss whether it satisfies the conditions for virtualization as defined in 1974.

3 Containerization

3.1 Basics

- Motivation: Trade isolation for efficiency (see [Sol+07])

    - – **Main idea** of containerization: **Share kernel** among containers
        - ∗ (Instead of separate OS per VM)

- Mechanisms

    - – Add container ID to each process, add new access control checks to system calls
    - – In case of Linux kernel
        - ∗ Kernel namespaces
            - · Limit what is visible inside container
        - ∗ Control groups (cgroups)
            - · Limit resource usage
        - ∗ Copy-on-write, e.g., UnionFS
            - · New container without copying all files, localized changes
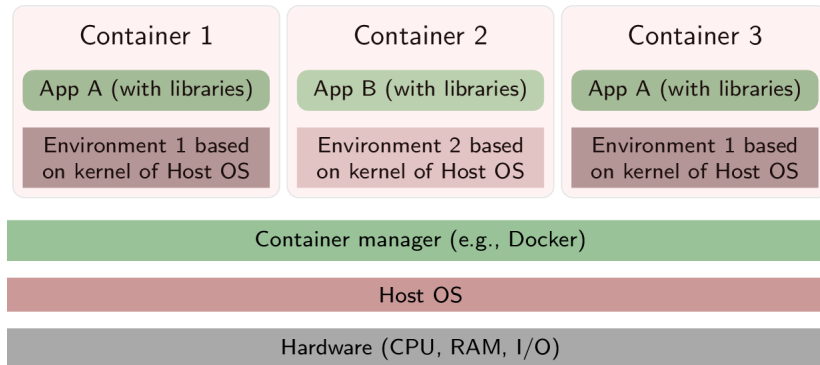
3.2 Layering with Containerization

Figure 4: Layering with containerization

3.3 Selected Technologies

- Docker

Figure 5: "Docker logo" under Docker Brand Guidelines; from Docker

  - **Image** describes OS/application environment: What software/configuration?
    * **Registries** publish images
    * Dockerfiles are build recipes for images in simple text format
  - **Container** is process (set), created from image (image is template for container)

- Kubernetes

Figure 6: "Kubernetes logo" under Kubernetes Branding Guidelines; from GitHub

  - Cluster manager for Docker
    * Pod = group of containers sharing resources, unit of deployment
    * Pods can be replicated (copied) for scalability
    * Integrated load-balancer

3.3.1   On Images

- With VMs, you could install software as in any other OS

  - Getting messy over time

- With Docker, images are defined via Dockerfiles

  - Explicitly listing necessary pieces and dependencies
  - Enforcing order and reproducibility
  - Sample dockerfile (used in the past to generate reveal.js presentations and PDF from org files):

```
FROM ubuntu
LABEL maintainer="Jens Lechtenbörger"
RUN apt-get update && apt-get --no-install-recommends install -y \
    ca-certificates emacs git \
    texlive-bibtex-extra texlive-fonts-recommended texlive-generic-recommended \
    texlive-latex-base texlive-latex-extra texlive-latex-recommended
COPY manage-packages.el /tmp/
```

3.4   Review Question

- Which conditions for virtualization as defined in 1974 does Docker satisfy?

4   Docker

4.1   Docker Installation

- Community Edition of Docker available for different OSs

  - See here for installation links

- Install on one of your machines, ideally on one that you can bring to (or access in) class

  - Your installation may come with a graphical user interface (GUI), which you do **not** need
    * Some students perceive the GUI to be confusing
    * Use command line instead to enter commands shown subsequently (any terminal should work, maybe try Bash)

4.2   First Steps

- Run `hello-world` as instructed in Get Started

  - In case of problems, please ask in the forum

- List your images and containers

  - `docker image ls`
  - `docker container ls -all`

* Help is available, e.g.:
    · `docker container --help`
    · `docker container ls --help`

- Maybe delete image and container

  – `docker rmi -f hello-world`

4.3   A Web Server

- Run nginx

  – `docker run -p 8080:80 nginx`
    * `-p`: Web server listens on port 80 in container; bind to port 8080 on host
        · Visit local server (see subsequent slide for Docker Toolbox under Windows)
    * Maybe add option `--name my-nginx`: Assign name to container for subsequent use
        · E.g., `docker stop/start/logs/rm my-nginx`

- Serve own HTML files

  – Add option `-v` in above `docker run ...` (**before** `nginx`)
    * Mount (make available) directory from host in container
        · E.g.: `-v /host-directory/with/html-files:/usr/share/nginx/html`
        · `/usr/share/nginx/html` is where nginx expects HTML files, in particular `index.html`
        · Thus, your HTML files replace default ones of nginx

4.3.1   Selected Errors

- **Error** message: name in use already

  – You cannot use the same name multiple times with `docker run --name ...`
  – Instead: `docker start my-nginx`

- **Error** message: port is allocated already

  – You cannot use option `-p` with same port in several `docker run` invocations
    * Other container still running, stop first
        · `docker ps`: Note ID or name
        · `docker stop <ID-or-name>`
        · `docker run ...`
    * (Or some other process uses that port. Kill process or choose different port.)

4.3.2 On Option `-v`

- Say, you start `nginx` with option `-v` but your files do not appear

  - `docker inspect <name-or-id-of-container>`

    * Check output for `binds`, telling you what is mapped to `/usr/share/nginx/html`
      · May not meet your expectations
  - Are you on Windows?

    * Try `-v C:\Users\...` with Powershell
    * Try `-v C:\\Users/...` with Bash
    * Try `-v /mnt/c/Users/...` with WSL terminal

4.3.3 Docker Toolbox under Windows

- (I do not recommend this in any way. Switch to GNU/Linux.)

- Docker Toolbox installs a virtual machine, in which Docker runs

  - Initial output informs about

    * IP address of VM, e.g., `192.168.99.100`
      · Visit port 8080 on 192.168.99.100
    * File system path
      · `/c/Program Files/Docker Toolbox`
  - Paths under `C:\Users` can be mounted by default

    * E.g., `docker run -p 8080:80 -v /c/Users/<your-name>/<folder-with-index.html>:/u`
      `nginx`
      · Maybe you need double slashes

5 Conclusions

5.1 Summary

- Virtual **virtual machines** are **efficient**, **isolated duplicates** of the real machine

- **Containers** are running processes, defined by **images**

  - Containers on one host share same OS kernel

- Virtual machines and containers

  - can be contrasted in terms of their layering approaches
  - allow to deploy software in well-defined environments

5.2 Outlook

- Containerization (in combination with version control such as offered by Git) is enabler of **DevOps**

  - DevOps = Combination of Development and Operations, see [Jab+16; Wie+19]

    * Bridge gaps between teams and responsibilities
    * Aiming for rapid software release cycles with high degree of automation and stability

  - Trend in software engineering

    * Communication and collaboration, continuous integration (CI) and continuous deployment (CD)
    * Approach based on Git also called GitOps, see [Lim18]
      · Self-service IT with proposals in pull requests (PRs)
      · Infrastructure as Code (IaC)

Bibliography

[BH07]   L. A. Barroso and U. Hölzle. "The Case for Energy-Proportional Computing". In: *Computer* 40.12 (2007), pp. 33–37. DOI: 10.1109/MC.2007.443. URL: https://www.barroso.org/publications/ieee_computer07.pdf.

[Cas+19]   Paul Castro et al. "The Rise of Serverless Computing". In: *Commun. ACM* 62.12 (Nov. 2019), pp. 44–54. DOI: 10.1145/3368454. URL: https://doi.org/10.1145/3368454.

[Fel+15]   Wes Felter et al. "An updated performance comparison of virtual machines and linux containers". In: *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE. 2015, pp. 171–172.

[Gol74]   Robert P. Goldberg. "Survey of virtual machine research". In: *Computer* 7.6 (1974), pp. 34–45.

[Jab+16]   Ramtin Jabbari et al. "What is DevOps? A Systematic Mapping Study on Definitions and Practices". In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. XP '16 Workshops. 2016. DOI: 10.1145/2962695.2962707. URL: https://doi.org/10.1145/2962695.2962707.

[Lim18]   Thomas A. Limoncelli. "GitOps: A Path to More Self-Service IT". In: *Commun. ACM* 61.9 (2018), pp. 38–42. DOI: 10.1145/3233241. URL: https://doi.org/10.1145/3233241.

[PG74]   Gerald J. Popek and Robert P. Goldberg. "Formal Requirements for Virtualizable Third Generation Architectures". In: *Commun. ACM* 17.7 (July 1974), pp. 412–421. ISSN: 0001-0782. DOI: 10.1145/361011.361073. URL: http://doi.acm.org/10.1145/361011.361073.

[Sol+07]   Stephen Soltesz et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors". In: *ACM SIGOPS Operating Systems Review*. Vol. 41. 3. ACM. 2007, pp. 275–287.

[Vog08]   Werner Vogels. "Beyond Server Consolidation: Server Consolidation Helps Companies Improve Resource Utilization, but Virtualization Can Help in Other Ways, Too." In: *Queue* 6.1 (2008), pp. 20–26. DOI: 10.1145/1348583.1348590. URL: https://doi.org/10.1145/1348583.1348590.

[Wie+19]   Anna Wiedemann et al. "Research for Practice: The DevOps Phenomenon". In: *Commun. ACM* 62.8 (2019), pp. 44–49. DOI: 10.1145/3331138. URL: https://doi.org/10.1145/3331138.

License Information

This document is part of a larger course. Source code and source files are available on GitLab under free licenses.