Big Data Analytics
Lecture 3: Virtualization and Containerization

Based on Slides by Dr. Tariq Mahmood

Fall 2025

Abstract

These notes provide a comprehensive overview of virtualization and containerization, two foundational technologies in modern computing and big data infrastructure. We begin with the history and core concepts of hardware virtualization, including the role of the hypervisor, and explore its various types such as server, desktop, storage, network, and data virtualization. The discussion covers key features like snapshots and migration. We then transition to containerization, a lightweight alternative, detailing its OS-level approach and its advantages in efficiency and performance. The notes conclude with a detailed comparison between virtualization (VMs) and containerization (e.g., Docker), highlighting their respective architectures, use cases, and trade-offs.

Contents

1 Introduction to Virtualization **2**
- 1.1 Historical Context . . . 2
- 1.2 A Timeline of Key Milestones . . . 2

2 Hardware Virtualization **2**
- 2.0.1 Core Terminology . . . 2

3 The Need for Virtualization **3**

4 Types of Virtualization **3**
- 4.1 Server Virtualization . . . 3
- 4.2 Desktop Virtualization (VDI) . . . 3
- 4.3 Storage Virtualization . . . 4
- 4.4 Network Virtualization . . . 4
- 4.5 Data Virtualization . . . 4
- 4.6 Application Virtualization . . . 4

5 Core Virtualization Features **4**
- 5.1 Snapshots . . . 4
- 5.2 Migration and Failover . . . 4

6 The Hypervisor in Depth **5**
- 6.1 Type 1 vs. Type 2 Hypervisors . . . 5
- 6.2 Advantages and Disadvantages of Virtualization . . . 5

7 The Shift to Containerization **6**
- 7.1 What is Containerization? . . . 6
- 7.2 Docker: The Face of Containerization . . . 6

8 Virtualization vs. Containerization: A Detailed Comparison **6**

1 Introduction to Virtualization

Virtualization is the process of creating a virtual, rather than actual, version of something, such as an operating system, a server, a storage device, or network resources.

1.1 Historical Context

The concept of virtualization dates back to the 1960s. It originated with mainframe computers, where the primary goal was to **logically divide the system resources** provided by a single, powerful mainframe between different applications. This allowed for time-sharing and more efficient use of expensive hardware. While the fundamental idea remains, the definition and application of virtualization have evolved significantly since then.

1.2 A Timeline of Key Milestones

The journey from mainframe time-sharing to modern cloud-native containerization is marked by several key innovations:

- **1966:** The term **Hypervisor** is used for the first time to describe the software that manages virtual machines.
- **1967:** IBM Cambridge develops **CP-40**, the first operating system to use complete virtualization.
- **1980s:** The availability of commodity **x86 servers** and desktop systems sets the stage for mainstream virtualization.
- **1999:** The release of **VMware Workstations** brings robust hardware virtualization to the x86 platform, making it widely accessible.
- **2003:** The open-source virtualization platform **Xen** is released, becoming a cornerstone of many cloud providers, including AWS.
- **2007: VirtualBox** is released, providing another popular open-source virtualization solution. In the Linux world, Process Containers are renamed to **CGroups** and merged into the Linux Kernel, a foundational technology for modern containers.
- **2008:** Microsoft launches its hypervisor, **Hyper-V**. The introduction of Linux containers and **namespaces** provides the kernel-level isolation needed for containerization.
- **2013:** The introduction of **Docker** revolutionizes the industry by making containerization easy to use, leading to its widespread adoption.
- **2014:** Google releases **Kubernetes**, an open-source container orchestration system for automating the deployment, scaling, and management of containerized applications.
- **2016:** Microsoft introduces native **Windows Containers**.

2 Hardware Virtualization

Hardware virtualization is the creation of a **virtual machine (VM)** that acts like a real computer with its own operating system. Software executed on these VMs is separated from the underlying physical hardware.

2.0.1 Core Terminology

Host Machine The physical machine on which the virtualization software runs.

Guest Machine The virtual machine itself. Software running inside the VM is said to be running on the guest machine.

Hypervisor Also known as a Virtual Machine Monitor (VMM), this is the software or firmware that creates and runs virtual machines. The hypervisor is responsible for abstracting and managing the host's physical hardware resources (CPU, memory, storage) and allocating them to the guest machines.

For example, a Windows computer (host machine) can run a hypervisor like VirtualBox to host a VM that looks like a computer with Ubuntu Linux (guest machine). Ubuntu-based software can then be run on this VM, completely isolated from the host's Windows OS.

3 The Need for Virtualization

Virtualization solves several fundamental problems in traditional IT infrastructure.

- **Resource Utilization:** In traditional IT, servers were often dedicated to a single application, leading to severe underutilization of hardware, with typical utilization rates as low as 10-15%. Virtualization allows multiple VMs to run on a single physical server, consolidating workloads and significantly improving resource utilization.
- **Cost Efficiency:** By consolidating workloads onto fewer physical machines, virtualization reduces hardware, energy, cooling, and maintenance costs.
- **Scalability and Flexibility:** Traditional servers are not flexible. Scaling required purchasing and configuring new hardware, which was slow and expensive. Virtualization allows VMs to be created, modified, or moved between physical servers in minutes, enabling a rapid response to changing demands.
- **Isolation and Security:** Running multiple applications on a single OS can lead to conflicts and security risks. Virtualization provides a strong layer of isolation between VMs. A failure, crash, or security breach in one VM does not affect other VMs running on the same physical hardware.

4 Types of Virtualization

Virtualization can be applied to many different layers of the IT stack.

4.1 Server Virtualization

This is the process of dividing one physical server into multiple isolated virtual servers (VMs) using a hypervisor. Each VM behaves like a complete, independent machine with its own:

- **Operating System (Guest OS):** Each VM runs its own full OS (e.g., Windows, Linux).
- **Virtual CPU (vCPU):** The hypervisor maps vCPUs to the host's physical CPU cores.
- **RAM:** A portion of the host's physical RAM is allocated to each VM.
- **Disk & Network:** Virtual disks and network interfaces are created and mapped to physical resources.

Benefit: Instead of dedicating one physical server per workload, you can run many workloads on one physical server, drastically improving hardware utilization. For example, a physical server with 32 cores and 128 GB RAM could host 10 web server VMs, 5 database VMs, and 2 analytics VMs, all running in isolation on the same hardware.

4.2 Desktop Virtualization (VDI)

This technology separates a user's desktop environment (the entire OS, apps, files, and settings) from the physical computer. The desktop runs in a virtualized environment on a central server, and users connect to it remotely from a client device (like a thin client, laptop, or tablet).

- **Benefits:** Anywhere access, centralized management (updates and patching are done in one place), enhanced security (data stays in the data center, not on the endpoint device), and cost-efficiency (users don't need powerful PCs).

4.3 Storage Virtualization

This is the process of pooling multiple physical storage devices (HDDs, SSDs, SAN/NAS systems) into a **single logical storage resource**. Applications and users can access this pool as if it were one big disk. A virtualization layer maps the logical storage requests to the appropriate physical storage device.

- **Benefits:** Simplified management from a single console, better utilization of storage resources, high availability through data mirroring, and flexibility to move data without disrupting applications.

4.4 Network Virtualization

This involves abstracting physical networking hardware (switches, routers, firewalls) into logical, flexible, software-defined virtual networks. This is a key component of **Software-Defined Networking (SDN)**, where the network's control logic is decoupled from the physical hardware and managed by a centralized controller.

- **Benefits:** Allows for the creation of complex network topologies in software, automates network provisioning, and enables services like virtual firewalls and load balancers to be run as software appliances (Virtual Network Functions or VNFs).

4.5 Data Virtualization

This technology allows you to access and query data from multiple, disparate sources as if it were a single database, **without physically moving or copying the data**. Instead of building costly ETL pipelines to replicate data into a central warehouse, data virtualization provides a real-time, unified view.

- **How it works:** A virtual data layer uses connectors to link to sources (databases, APIs, files). When a query is made, the virtualization engine translates it, fetches the required data from the sources in real time, and presents the results as if they came from one unified system.
- **Example:** A bank can get a 360-degree view of a customer by creating a virtual profile that queries transactions from an Oracle DB, credit reports from an API, and support tickets from Salesforce, all on-demand without data replication.

4.6 Application Virtualization

This technique runs an application in an isolated virtual environment on a client machine, rather than installing it directly on the underlying OS. A virtualization layer intercepts the app's calls to the OS (e.g., for file or registry access) and redirects them to a virtualized container.

- **Benefits:** Prevents conflicts between applications (no "DLL Hell"), simplifies deployment, enhances security (app changes don't affect the host OS), and improves portability.
- **Example:** Running two different versions of Microsoft Word, like Word 2010 and Word 2021, on the same PC without them interfering with each other.

5 Core Virtualization Features

5.1 Snapshots

A snapshot captures the state of a virtual machine—including its memory, settings, and storage devices—at an exact point in time. It allows the VM's state to be restored to that point later, effectively undoing any subsequent changes. This is extremely useful as a backup technique before performing risky operations like software updates or configuration changes. Snapshots work by preserving the original virtual disk and writing all subsequent changes to a separate delta disk file.

5.2 Migration and Failover

Migration The process of moving a running virtual machine from one physical host to another with minimal or no downtime. **Live migration** moves the VM's memory state over the network while the VM continues to run, ensuring a seamless transition. This is essential for load balancing and performing hardware maintenance without service interruption.

Failover An automated process that allows a VM to continue operations if its host fails. In a clustered environment, if a host machine goes down, a failover mechanism automatically restarts the VMs that were running on it on another healthy host in the cluster.

6 The Hypervisor in Depth

The hypervisor is the core software that enables virtualization. There are two main types.

6.1 Type 1 vs. Type 2 Hypervisors

Type 1 (Bare-Metal) Hypervisor Runs directly on the host's physical hardware to control the hardware and manage guest operating systems. It acts as the operating system for the machine. This type offers higher performance and security as there is no middle OS layer.

- **Examples:** VMware ESXi, Microsoft Hyper-V, Citrix XenServer, KVM.

Type 2 (Hosted) Hypervisor Runs as a software application on top of a conventional host operating system (like Windows or macOS). It is simpler to set up but has higher overhead and lower performance because it has to go through the host OS to access hardware resources.

- **Examples:** VMware Workstation, Oracle VirtualBox, Parallels Desktop.

tableheader		
AKA	Bare-metal or Native	Hosted
Definition	Runs directly on the system hardware with VMs running on them.	Runs as an application on a conventional Operating System.
Performance	Higher performance as there is no middle OS layer to add overhead.	Comparatively reduced performance as it runs with the extra overhead of the host OS.
Security	More secure. Isolation is handled at a lower level.	Less secure, as any security problem in the host OS can potentially affect the hypervisor and all guest VMs.
Setup	More complex, often requires dedicated hardware.	Lot simpler setup, as you already have a host OS.
Examples	VMware ESXi, Microsoft Hyper-V, Citrix XenServer	VMware Workstation, Oracle VirtualBox

6.2 Advantages and Disadvantages of Virtualization

Advantages:

- Enables multiple operating systems to run on the same machine.
- Cheaper due to less physical hardware and a more compact infrastructure setup.
- Easy to recover from failure using snapshots and migration.
- Faster provisioning of new servers and applications.
- Increased IT productivity and simplified management.

Disadvantages:

- Running many VMs on a single host can lead to resource contention and unstable performance if not managed properly.
- A hypervisor introduces an overhead layer, making it inherently less efficient than running directly on the host operating system.
- Each VM requires a full guest OS, which consumes significant processor, disk, and RAM resources.
- The boot-up process for a full VM is long, taking minutes.

7 The Shift to Containerization

Containerization addresses many of the disadvantages of traditional hypervisor-based virtualization by bringing virtualization to the operating system level.

7.1 What is Containerization?

Instead of virtualizing the entire hardware stack, containerization virtualizes the **operating system resources**.

- All containers running on a host share the same **host OS kernel**.
- A container is an isolated, lightweight silo that contains an application and all its required binaries and libraries.
- Because there is no guest OS to boot, containers start in seconds and have minimal performance overhead.
- This lightweight approach allows for much higher density—you can run many more containers than VMs on the same hardware.

Any application can be bundled in a container and run without worrying about dependencies, libraries, and binaries, ensuring that it works consistently across different environments (e.g., from a developer's laptop to a production server).

7.2 Docker: The Face of Containerization

Docker is a containerization platform that packages an application and its dependencies together in the form of a container.

- **Consistency:** Docker ensures that the working environment is consistent for everyone involved in the software lifecycle, from development to testing and deployment. A developer can build a container, and the QA team only needs to run that container to replicate the exact developer environment.
- **Process-Level Isolation:** Each app runs in a separate container, independent of others, preventing interference.
- **Efficiency:** The QA team doesn't need to install all dependent software; they just run the container, saving time and energy.
- **Scalability:** Systems can be scaled up easily by launching more instances of a container, and code can be deployed effortlessly.

8 Virtualization vs. Containerization: A Detailed Comparison

While both provide isolation, their approach and trade-offs are fundamentally different.

tableheader		
Framework	Virtualization at the hardware level. Each VM includes a full copy of an OS.	Virtualization at the OS level. Containers share the host OS kernel.
Segregation	Full segregation. VMs are completely isolated from each other and the host.	Process or application-level segregation. Containers are isolated but share the same kernel.
Portability	Less portable due to hardware and OS dependencies. Moving a VM involves moving a large virtual disk.	Highly portable because of its lightweight nature and no dependence on the underlying hardware or a specific OS kernel version (within the same OS family).
Size	Heavyweight, typically gigabytes in size.	Lightweight, typically megabytes in size.
Boot Time	Minutes.	Seconds.
Security	More secure due to full hardware-level isolation. A compromised VM does not affect the host or other VMs.	Less secure in comparison. A severe kernel vulnerability in the host OS could potentially affect all containers.

References

[1] VMware. "What is Virtualization?". https://www.vmware.com/solutions/virtualization.html

[2] Docker Inc. "What is a Container?". https://www.docker.com/resources/what-container/