T-401-ICYB
Virtualization

Stephan Schiffel
stephans@ru.is
Reykjavik University, Iceland

25.11.2025

Outline

What is Virtualization

What is Virtualization? – The Core Concept

"Virtualization (abbreviated v12n) is a series of technologies that allows dividing of physical computing resources into a series of virtual machines, operating systems, processes or containers."

`https://en.wikipedia.org/wiki/Virtualization`

- **Key Idea:** Abstracting physical resources, allowing them to be shared or replicated in software.
- **Analogy:** Think of a large apartment building (physical server) divided into many individual apartments (virtual machines), each with its own utilities and inhabitants, but sharing the same underlying structure.

Purpose

Why Virtualize? – Doing More with Less, More Flexibly

- **Resource Utilization:** Maximize the use of expensive physical hardware by running multiple "virtual" instances on a single machine.
- **Cost Savings:** Reduce hardware purchases, power consumption, cooling costs, and data center space.
- **Flexibility & Agility:** Rapidly provision, deploy, and scale resources as needed, significantly faster than deploying new physical hardware.
- **Isolation & Security:** Virtual environments are isolated from each other, preventing issues in one from affecting others and providing a sandbox for sensitive operations.
- **Business Continuity & Disaster Recovery:** Easier to back up, replicate, and restore entire systems, improving resilience against failures.

Common Uses of Virtualization Today

- **Server Consolidation:** Running dozens of virtual servers on a single physical server, reducing hardware footprint.
- **Development & Testing Environments:** Quickly spinning up isolated VMs for testing software without affecting production systems.
- **Cloud Computing (IaaS):** The foundational technology behind services like AWS EC2, Azure VMs, Google Cloud Compute Engine.
- **Desktop Virtualization:** Delivering virtual desktops to users, allowing access from any device (VDI - Virtual Desktop Infrastructure).
- **Disaster Recovery & Business Continuity:** Replicating and restoring entire virtualized environments with ease.
- **Legacy Application Support:** Running older operating systems or applications that require specific hardware/environments.
- **Security Sandboxing:** Safely running untrusted applications or browsing the web in an isolated environment.

Terminology

Who's Who in the Virtual World

- **Host Machine/OS:**
  - The physical computer on which virtualization software runs.
  - The operating system (if any) installed directly on the physical hardware.
- **Guest Machine/OS (Virtual Machine, VM):**
  - A software-based, isolated computer system that runs on the host.
  - Has its own virtual hardware (CPU, RAM, disk, network) and operating system.
- **Hypervisor (Virtual Machine Monitor, VMM):**
  - The software layer that creates, runs, and manages virtual machines.
  - Acts as a mediator between the virtual machines and the physical hardware.

Types of Hypervisors

- **Type 1 Hypervisor (Bare-Metal Hypervisor):**
  - Runs directly on the host hardware, without an underlying operating system.
  - **Pros:** High performance, greater security, enterprise-grade.
  - **Examples:** VMware ESXi, Microsoft Hyper-V, KVM, Citrix XenServer.
- **Type 2 Hypervisor (Hosted Hypervisor):**
  - Runs as an application on top of a conventional host operating system.
  - **Pros:** Easier to set up, good for desktop development/testing.
  - **Examples:** Oracle VirtualBox, VMware Workstation

Virtualization Methods

Virtualization Methods

- **Full Virtualization:** Hypervisor fully simulates hardware, guest OS runs unmodified.
- **Paravirtualization:** Guest OS is modified/aware it is virtualized, communicates directly with hypervisor.
- **Hardware-Assisted Virtualization:** Leverages special CPU instructions (Intel VT-x, AMD-V) to accelerate virtualization.
- **OS-Level Virtualization (Containerization):** Shares the host OS kernel but isolates applications in "containers."
- **Emulation:** Simulates an entire hardware platform, often for different CPU architectures.

Full Virtualization

- **How it Works:** The hypervisor fully simulates all hardware components for the guest OS. The guest OS runs unmodified, believing it has direct access to physical hardware.
- **Hardware-Assisted Virtualization (e.g., Intel VT-x, AMD-V):** Modern CPUs include features that offload much of the virtualization burden from the hypervisor, significantly improving performance.

  Pros:

  - Guest OS runs unmodified: High compatibility with most operating systems.
  - Strong isolation: VMs are highly isolated from each other and the host.
  - Flexibility: Can run different OS families (e.g., Windows on a Linux host).
  - Near-native performance with hardware assistance.

  Cons:

  - Higher overhead without hardware assistance (due to full simulation).
  - Can be resource-intensive (CPU, RAM) compared to lighter methods.
  - Requires CPU support for hardware-assisted features for optimal speed.

Paravirtualization / Hybrid virtualization

- **How it Works:** The guest operating system is modified (or uses special drivers) to be "aware" it's running in a virtualized environment. It makes special "hypercalls" to the hypervisor instead of directly issuing hardware instructions.
- **Key Idea:** Guest OS and hypervisor cooperate to achieve better performance.
- **Pros:**
  - Lower overhead and better performance than pure full virtualization (without hardware assistance).
  - More efficient use of resources.
- **Cons:**
  - Requires modification or special drivers for the guest OS kernel, reducing compatibility.
  - Can be more complex to set up due to guest OS modifications.
- Less common as a primary method now that hardware-assisted full virtualization is widely available and performant.

OS-Level Virtualization (Containers)

- = Shared Kernel, Isolated User Space
- **How it Works:** Containers share the host operating system's kernel, device drivers, etc). Each container has its own isolated user space, processes, file system, and network interfaces.
- **Key Idea:** Focuses on isolating applications and their dependencies, not entire operating systems.
- **Pros:**
  - Extremely lightweight and fast startup (seconds to milliseconds).
  - Very efficient resource utilization (no separate OS kernel per container).
  - Highly portable: "Build once, run anywhere" on compatible host OS.
  - Ideal for microservices, continuous integration/delivery (CI/CD).
- **Cons:**
  - Less isolation than VMs: Shared kernel is a single point of failure/vulnerability.
  - All containers must run on the same host OS kernel (e.g., Linux containers on Linux only).

Emulation - Simulating a Different System

- **How it Works:** Completely simulates the hardware and instruction set of a different system (often a different CPU architecture). The guest code is translated on the fly to run on the host's native architecture.
- **Key Idea:** Allows software designed for one type of hardware to run on entirely different hardware.
- **Pros:**
  - Runs software for incompatible hardware architectures (e.g., ARM software on an x86 machine).
  - The guest software needs no modification.
  - Useful for running legacy software or retro games.
- **Cons:**
  - Very high performance overhead: Significantly slower than virtualization due to constant instruction translation.
  - Highly resource-intensive, especially for CPU.
  - Primarily used when virtualization is not an option due to fundamental architectural differences.
- Not typically used for general-purpose server consolidation or cloud infrastructure due to performance.

Security Issues

Virtualization Security: A New Layer of Risk

- Virtualization abstracts physical hardware, but introduces new attack vectors.
- Multiple workloads (VMs/containers) share a single physical host.
- A security breach can have a wider impact than on isolated physical machines.
- Understanding these risks is crucial for secure infrastructure.

Common Virtualization Security Challenges

- **VM/Container Escape:**
  - Breaking out of a guest to compromise the host/hypervisor.
  - The "holy grail" for attackers in virtualized environments.
- **Hypervisor Vulnerabilities:**
  - Flaws in the core virtualization software can affect ALL guests.
  - Critical single point of failure.
- **Resource Exhaustion (DoS):**
  - One guest consuming excessive host resources, starving others.
- **Management Plane Security:**
  - Compromise of virtualization management tools gives full control.
- **Insecure Images/Templates:**
  - Deploying guests with known vulnerabilities or misconfigurations.

Security by Method: Full Virtualization - Bare-Metal Hypervisors

- **Architecture:** Hypervisor runs directly on hardware. Guests are fully isolated with virtual hardware.
- **Key Risks:** Hypervisor vulnerabilities, VM escape.

Pros:

- **Strong Isolation:** Hardware-level separation between VMs.
- **Small Attack Surface:** Hypervisor is purpose-built, lean.
- Independent Guest OSes.

Cons:

- Hypervisor bug impacts ALL VMs.
- Management tools are high-value targets.

Security by Method: Full Virtualization - Hosted Hypervisors

- **Architecture:** Hypervisor runs as an application on a host OS (e.g., Windows, Linux).
- **Key Risks:** Host OS vulnerabilities, VM escape, hypervisor application flaws.

Pros:

- Good isolation *between VMs.*
- Leverage host OS security tools (AV, firewall).

Cons:

- **Larger Attack Surface:** Host OS + Hypervisor application.
- Host OS vulnerabilities directly impact guests.
- Weaker overall isolation than Type 1.

Security by Method: OS-Level Virtualization (Containers)

- **Architecture:** Containers share the host OS kernel.
- **Key Risks:** Shared kernel vulnerabilities, container escape, insecure images, orchestration flaws.

Pros:

- Smaller per-container attack surface (no full OS).
- Rapid patching/deployment of immutable images.

Cons:

- **Weaker Isolation:** Shared kernel is a single point of failure.
- Kernel exploit affects ALL containers.
- High reliance on secure images.

Security by Method: Emulation

- **Architecture:** Simulates entire hardware/instruction set (often for different CPU arch).
- **Key Risks:** Emulator software vulnerabilities, high performance overhead.

Pros:

- Strong isolation *from host* due to full translation.

Cons:

- Not designed for secure, multi-tenant production.
- High performance overhead.
- Vulnerabilities in emulator can be exploited by guest.

Comparative Security: Which is "More Secure"?

1. **Physical Hardware:** Ultimate isolation, but inefficient.
2. **Full Virtualization (Type 1):**
   - **Highest Isolation** among common virtualization methods.
   - Best for multi-tenancy, cloud providers.
   - Minimal hypervisor attack surface.
3. **Full Virtualization (Type 2):**
   - Less secure due to reliance on underlying host OS.
   - Larger attack surface.
4. **OS-Level Virtualization (Containers):**
   - **Least Isolation** among common virtualization methods.
   - Shared kernel is a critical security trade-off for efficiency.
   - Requires strong host OS security, image hygiene, and orchestration controls.
5. **Emulation:** Niche use, not typically for general infrastructure security.

Mitigating Virtualization Security Risks

- **Patch Management:** Keep hypervisors, hosts, guests, and containers up-to-date.
- **Secure Configuration:**
  - Apply principle of least privilege.
  - Disable unused features/ports.
  - Enforce strong access controls.
- **Network Segmentation:** Isolate management networks, guest networks, and host networks.
- **Secure Image/Template Management:** Scan for vulnerabilities, use trusted sources.
- **Monitoring & Logging:** Detect unusual activity, VM escapes, or resource abuse.
- **Guest Hardening:** Apply security best practices within each VM/container.
- **Hypervisor/Host Hardening:** Follow vendor guidelines, remove unnecessary software.

Up Next ..

Further Studies

- Under what specific scenarios would you have to choose VMs over containers for security reasons, and vice-versa?
- Research historical examples of VM escape vulnerabilities (e.g., specific CVEs for VMware, VirtualBox, KVM, or Docker). What were the root causes?
- Side-channel attacks:
  - What is a side-channel attack (in the context of virtualization)?
  - Why is this especially problematic for cloud environments?
  - Find recent examples of such attacks and discuss their impact.
  - What mitigation strategies exist to counter these types of attacks?
- What are the risks associated with using untrusted or vulnerable container images from public repositories?

Lab 2 - Virtual Machine

- Setup a hypervisor and run a virtual machine
- Should be fairly simple, giving you more time to do research.

Questions & Discussion

Please feel free to ask any questions or share your thoughts!