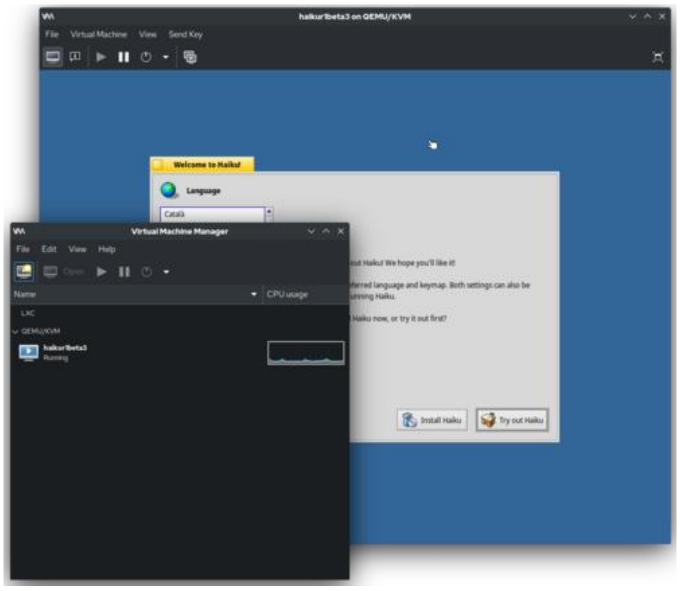
Virtual Machines(VM)

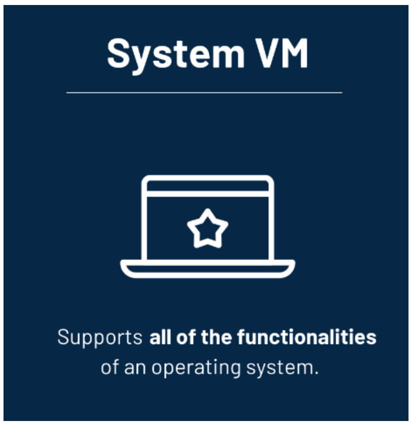


In computing, a virtual machine (VM) is the virtualization or emulation of a computer system. Virtual machines are based on computer architectures and provide the functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination of the two. Virtual machines differ and are organized by their function. System virtual machines (also called full virtualization VMs, or SysVMs) provide a substitute for a real machine. They provide the functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments that are isolated from one another yet exist on the same physical machine. Modern hypervisors use hardware-assisted virtualization, with virtualization-specific hardware features on the host CPUs providing assistance to hypervisors.

Process virtual machines are designed to execute computer programs in a platform-independent environment. Some virtual machine emulators, such as QEMU and video game console emulators, are designed to also emulate (or "virtually imitate") different system architectures, thus allowing execution of software applications and operating systems written for another CPU or architecture. OS-level virtualization allows the resources of a computer to be partitioned via the kernel. The terms are not universally interchangeable

System virtual machines

A 'virtual machine' was originally defined by Popek and Goldberg as "an efficient, isolated duplicate of a real computer machine." Current use includes virtual machines that have no direct correspondence to any real hardware. The physical, "real-world" hardware running the VM is generally referred to as the 'host', and the virtual machine emulated on that machine is generally referred to as the 'guest'. A host can emulate several guests, each of which can emulate different operating systems and hardware platforms.


System VM
Supports **all of the functionalities** of an operating system.

The desire to run multiple operating systems was the initial motive for virtual machines, so as to allow time-sharing among several single-tasking operating systems. In some respects, a system virtual machine can be considered a generalization of the concept of virtual memory that historically preceded it. IBM's CP/CMS, the first systems to allow full virtualization, implemented time sharing by providing each user with a single-user operating system, the Conversational Monitor System (CMS). Unlike virtual memory, a system virtual machine entitled the user to write privileged instructions in their code. This approach had certain advantages, such as adding input/output devices not allowed by the standard system.

Process virtual machines

A process virtual machine, sometimes called an application virtual machine, or Managed Runtime Environment (MRE), runs as a normal application inside a host OS and supports a single process. It is created when that process is started and deleted when it is closed. Its purpose is to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system and allows a program to execute in the same way on any platform.

Process VM

Supports only **one single process** at a time.

A process VM provides a high-level abstraction – that of a high-level programming language (compared to the low-level ISA abstraction of the system VM). Process VMs are implemented using an interpreter; performance comparable to compiled programming languages can be achieved by the use of just-in-time compilation. This type of VM has become popular with the Java programming language, which is implemented using the Java virtual machine. Other examples include the Parrot virtual machine and the .NET Framework, which runs on a VM called the Common Language Runtime. All of them can serve as an abstraction layer for any computer language.

Virtualization techniques

Full virtualization

In full virtualization, the virtual machine simulates enough hardware to allow an unmodified "guest" OS (one designed for the same instruction set) to be run in isolation. This approach was pioneered in 1966 with the IBM CP-40 and CP-67, predecessors of the VM family.

Hardware-assited virtualization

In hardware-assisted virtualization, the hardware provides architectural support that facilitates building a virtual machine monitor and allows guest OSes to be run in isolation. Hardware-assisted virtualization was first introduced on the IBM System/370 in 1972, for use with VM/370, the first virtual machine operating system offered by IBM as an official product. In 2005 and 2006, Intel and AMD provided additional hardware to support virtualization. Sun Microsystems (now Oracle Corporation) added similar features in their UltraSPARC T-Series processors in 2005. In 2006, first-generation 32- and 64-bit x86 hardware support was found to rarely offer performance advantages over software virtualization.

OS-level virtualization

In OS-level virtualization, a physical server is virtualized at the operating system level, enabling multiple isolated and secure virtualized servers to run on a single physical server. The "guest" operating system environments share the same running instance of the operating system as the host system. Thus, the same operating system kernel is also used to implement the "guest" environments, and applications running in a given "guest" environment view it as a stand-alone system. The pioneer implementation was FreeBSD jails; other examples include Docker, Solaris Containers, OpenVZ, Linux-VServer, LXC, AIX Workload Partitions, Parallels Virtuozzo Containers, and iCore Virtual Accounts.

Nested virtualization

Nested virtualization refers to the ability of running a virtual machine within another, having this general concept extendable to an arbitrary depth. In other words, nested virtualization refers to running one or more hypervisors inside another hypervisor. The nature of a nested guest virtual machine does not need to be homogeneous with its host virtual machine; for example, application virtualization can be deployed within a virtual machine created by using hardware virtualization. Nested virtualization becomes more necessary as widespread operating systems gain built-in hypervisor functionality, which in a virtualized environment can be used only if the surrounding hypervisor supports nested virtualization; for example, Windows 7 is capable of running Windows XP applications inside a built-in virtual machine. Furthermore, moving already existing virtualized environments into a cloud, following the Infrastructure as a Service (IaaS) approach, is much more complicated if the destination IaaS platform does not support nested virtualization. The way nested virtualization can be implemented on a particular computer architecture depends on supported hardware-assisted virtualization capabilities. If a particular architecture does not provide hardware support required for nested virtualization, various software techniques are employed to enable it. Over time, more architectures gain required hardware support.