**OS-level virtualization** is an [operating system](https://en.wikipedia.org/wiki/Operating_system) paradigm in which the [kernel](https://en.wikipedia.org/wiki/Kernel_(computer_science)) allows the existence of multiple isolated [user space](https://en.wikipedia.org/wiki/User_space) instances. Such instances, called **containers** ([LXC](https://en.wikipedia.org/wiki/LXC), [Solaris containers](https://en.wikipedia.org/wiki/Solaris_Containers), [Docker](https://en.wikipedia.org/wiki/Docker_(software))), **Zones** ([Solaris containers](https://en.wikipedia.org/wiki/Solaris_Containers)), **virtual private servers** ([OpenVZ](https://en.wikipedia.org/wiki/OpenVZ" \o "OpenVZ)), **partitions**, **virtual environments** (VEs), **virtual kernels** ([DragonFly BSD](https://en.wikipedia.org/wiki/Vkernel)), or **jails** ([FreeBSD jail](https://en.wikipedia.org/wiki/FreeBSD_jail) or [chroot jail](https://en.wikipedia.org/wiki/Chroot_jail)),[[1]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-1) may look like real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can see all resources (connected devices, files and folders, [network shares](https://en.wikipedia.org/wiki/Shared_resource), CPU power, quantifiable hardware capabilities) of that computer. However, programs running inside of a container can only see the container's contents and devices assigned to the container.

On [Unix-like](https://en.wikipedia.org/wiki/Unix-like) operating systems, this feature can be seen as an advanced implementation of the standard [chroot](https://en.wikipedia.org/wiki/Chroot) mechanism, which changes the apparent root folder for the current running process and its children. In addition to isolation mechanisms, the kernel often provides [resource-management](https://en.wikipedia.org/wiki/Resource_management_(computing)) features to limit the impact of one container's activities on other containers.

The term *container*, while most popularly referring to OS-level virtualization systems, is sometimes ambiguously used to refer to fuller [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) environments operating in varying degrees of concert with the host OS, e.g. [Microsoft's](https://en.wikipedia.org/wiki/Microsoft) [*Hyper-V*](https://en.wikipedia.org/wiki/Hyper-V)*containers*.

## Operation[[edit](https://en.wikipedia.org/w/index.php?title=OS-level_virtualization&action=edit&section=1)]

On ordinary operating systems for personal computers, a computer program can see (even though it might not be able to access) all the system's resources. They include:

1. Hardware capabilities that can be employed, such as the [CPU](https://en.wikipedia.org/wiki/CPU) and the network connection
2. Data that can be read or written, such as files, folders and [network shares](https://en.wikipedia.org/wiki/Network_share)
3. Connected [peripherals](https://en.wikipedia.org/wiki/Computer_peripheral) it can interact with, such as [webcam](https://en.wikipedia.org/wiki/Webcam), printer, scanner, or fax

The operating system may be able to allow or deny access to such resources based on which program requests them and the [user account](https://en.wikipedia.org/wiki/User_account) in the context of which it runs. The operating system may also hide those resources, so that when the computer program enumerates them, they do not appear in the enumeration results. Nevertheless, from a programming point of view, the computer program has interacted with those resources and the operating system has managed an act of interaction.

With operating-system-virtualization, or containerization, it is possible to run programs within containers, to which only parts of these resources are allocated. A program expecting to see the whole computer, once run inside a container, can only see the allocated resources and believes them to be all that is available. Several containers can be created on each operating system, to each of which a subset of the computer's resources is allocated. Each container may contain any number of computer programs. These programs may run concurrently or separately, and may even interact with one another.

Containerization has similarities to [application virtualization](https://en.wikipedia.org/wiki/Application_virtualization): In the latter, only one computer program is placed in an isolated container and the isolation applies to file system only.

## Uses[[edit](https://en.wikipedia.org/w/index.php?title=OS-level_virtualization&action=edit&section=2)]

Operating-system-level virtualization is commonly used in [virtual hosting](https://en.wikipedia.org/wiki/Virtual_machine) environments, where it is useful for securely allocating finite hardware resources among a large number of mutually-distrusting users. System administrators may also use it for consolidating server hardware by moving services on separate hosts into containers on the one server.

Other typical scenarios include separating several programs to separate containers for improved security, hardware independence, and added resource management features. The improved security provided by the use of a chroot mechanism, however, is nowhere near ironclad.[[2]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-2) Operating-system-level virtualization implementations capable of [live migration](https://en.wikipedia.org/wiki/Live_migration) can also be used for dynamic load balancing of containers between nodes in a cluster.

### Overhead**[**[**edit**](https://en.wikipedia.org/w/index.php?title=OS-level_virtualization&action=edit&section=3)**]**

Operating-system-level virtualization usually imposes less overhead than [full virtualization](https://en.wikipedia.org/wiki/Full_virtualization) because programs in OS-level virtual partitions use the operating system's normal [system call](https://en.wikipedia.org/wiki/System_call) interface and do not need to be subjected to [emulation](https://en.wikipedia.org/wiki/Emulator) or be run in an intermediate [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine), as is the case with full virtualization (such as [VMware ESXi](https://en.wikipedia.org/wiki/VMware_ESXi), [QEMU](https://en.wikipedia.org/wiki/QEMU), or [Hyper-V](https://en.wikipedia.org/wiki/Hyper-V)) and [paravirtualization](https://en.wikipedia.org/wiki/Paravirtualization) (such as [Xen](https://en.wikipedia.org/wiki/Xen) or [User-mode Linux](https://en.wikipedia.org/wiki/User-mode_Linux)). This form of virtualization also does not require hardware support for efficient performance.

### Flexibility**[**[**edit**](https://en.wikipedia.org/w/index.php?title=OS-level_virtualization&action=edit&section=4)**]**

Operating-system-level virtualization is not as flexible as other virtualization approaches since it cannot host a guest operating system different from the host one, or a different guest kernel. For example, with [Linux](https://en.wikipedia.org/wiki/Linux), different distributions are fine, but other operating systems such as Windows cannot be hosted. Operating systems using variable input systematics are subject to limitations within the virtualized architecture. Adaptation methods including cloud-server relay analytics maintain the OS-level virtual environment within these applications.[[3]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-3)

[Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)) partially overcomes the limitation described above with its [branded zones](https://en.wikipedia.org/wiki/Branded_zones) feature, which provides the ability to run an environment within a container that emulates an older [Solaris 8](https://en.wikipedia.org/wiki/Solaris_8) or 9 version in a Solaris 10 host. Linux branded zones (referred to as "lx" branded zones) are also available on [x86](https://en.wikipedia.org/wiki/X86)-based Solaris systems, providing a complete Linux [userspace](https://en.wikipedia.org/wiki/Userspace" \o "Userspace) and support for the execution of Linux applications; additionally, Solaris provides utilities needed to install [Red Hat Enterprise Linux](https://en.wikipedia.org/wiki/Red_Hat_Enterprise_Linux) 3.x or [CentOS](https://en.wikipedia.org/wiki/CentOS) 3.x [Linux distributions](https://en.wikipedia.org/wiki/Linux_distributions) inside "lx" zones.[[4]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-4)[[5]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-5) However, in 2010 Linux branded zones were removed from Solaris; in 2014 they were reintroduced in [Illumos](https://en.wikipedia.org/wiki/Illumos" \o "Illumos), which is the open source Solaris fork, supporting 32-bit [Linux kernels](https://en.wikipedia.org/wiki/Linux_kernel).[[6]](https://en.wikipedia.org/wiki/OS-level_virtualization#cite_note-6)

### Storage**[**[**edit**](https://en.wikipedia.org/w/index.php?title=OS-level_virtualization&action=edit&section=5)**]**

Some implementations provide file-level [copy-on-write](https://en.wikipedia.org/wiki/Copy-on-write) (CoW) mechanisms. (Most commonly, a standard file system is shared between partitions, and those partitions that change the files automatically create their own copies.) This is easier to back up, more space-efficient and simpler to cache than the block-level copy-on-write schemes common on whole-system virtualizers. Whole-system virtualizers, however, can work with non-native file systems and create and roll back snapshots of the entire system state.