TOPICS COVERED:

VIRTUALIZATION

- CLOUD DEPLOYMENT MODELS
- HYPERVISOR TYPE 1 & 2
- COMPUTE VIRTUALIZATION (Containers and Virtual machines)
- INTRO TO CLOUD COMPUTING
- CISCO SWITCH VIRTUALIZATION
  - Virtual Network Function (VNF)
  - Cisco Stackwise
  - Cisco 6500 VSS

CLOUD DEPLOYMENT MODELS

According to the definition of the National Institute of Standards and Technology (NIST) [https://www.nist.gov/] there are four cloud deployment models:

1. Public cloud
2. Private cloud
3. Community cloud
4. Hybrid cloud

1. Public Cloud

The public cloud is the cloud that most people are familiar with. The cloud infrastructure is available to everyone over the Internet. The cloud service provider (CSP) owns, manages, and operates the public cloud. The infrastructure is at the premises of the CSP.

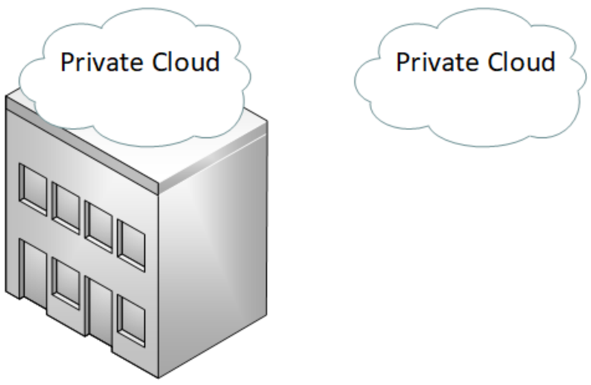
You pay for the services, storage, or compute resources you use. Customers that use the public cloud use shared resources. One advantage of the public cloud is that you don't need to buy and maintain the physical infrastructure. Your connection to the public cloud could be over the Internet or a private WAN connection.

Examples of public cloud providers:

- Amazon AWS
- Microsoft Azure
- Google Cloud
- IBM cloud
- Alibaba Cloud

2. Private Cloud

The private cloud is a cloud model where a **single organization uses the cloud**. The organization or a third party could own, manage, and operate the cloud. A combination of the two is also possible. This cloud can exist on-premises or off-premises.



Organizations that want more control over their cloud or that are bound by the law often use private clouds.

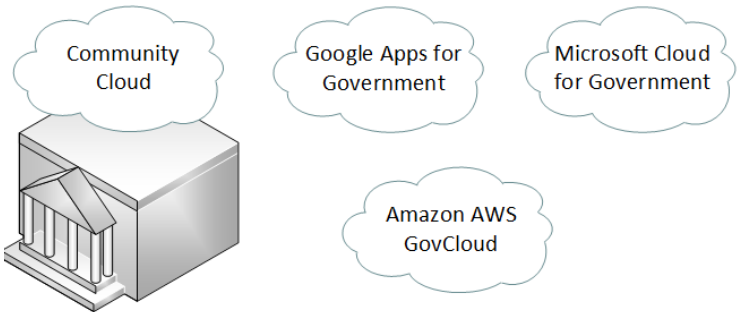
Here are four examples:

- OpenStack
- Microsoft Azure Stack
- VMWare vCloud Suite
- Amazon AWS Outposts

Public cloud providers can also emulate a private cloud within a public cloud. We call this a **virtual private cloud**. Amazon AWS and Google Cloud call this **Virtual Private Cloud (VPC).** Microsoft Azure calls this **Virtual Network (Vnet)**. VPC and Vnet isolate resources with a virtual network unreachable from other customers.

3. Community Cloud

The community cloud is a private cloud for organizations that **share common interests.** For example, mission objectives, compliance policies, and security. This cloud can exist on-premises or off-premises.



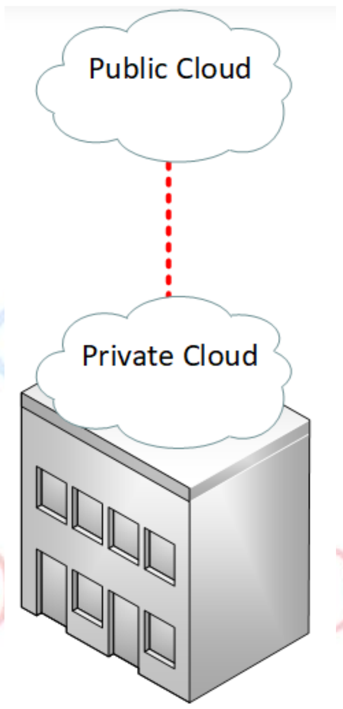
Because of regulatory standards and the law, the government and public sector often use community clouds. There are cloud providers that offer community clouds, here are some examples:

- Amazon AWS GovCloud
- Google Apps for Government

- Microsoft Cloud for Government
- Carpathia

4. Hybrid Cloud
A hybrid cloud is when we **combine a private and public cloud**. You don't have a hybrid cloud when you use a separate private and public cloud. We call it a hybrid cloud when the **private and public cloud is integrated**. An organization might run a Microsoft Exchange server in their private cloud but use Microsoft Azure Active Directory in the public cloud for authentication.

Another example is an organization that runs most of their workloads on their private cloud. When the private cloud is at 100% capacity, they use the public cloud. We call this **cloud bursting**. The scaling from the private cloud to the public cloud is seamless, you don't even notice whether you are on the private or public cloud.

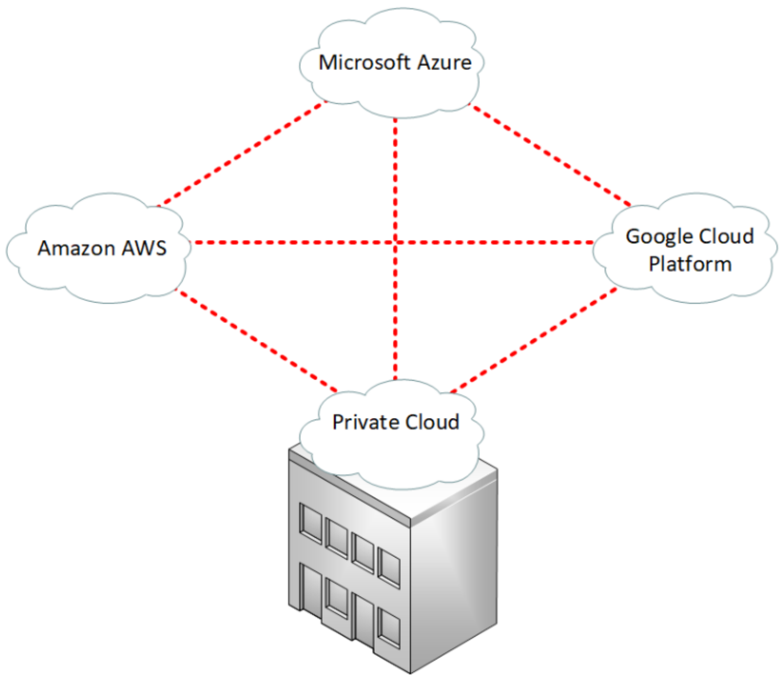


Here are three examples of hybrid clouds:

- Cisco Hybrid Cloud Platform for Google Cloud
- IBM Hybrid Cloud Platform
- Rackspace Hybrid Cloud

5. Multicloud
This cloud "model" is not in the NIST list of deployment models but good to know since many organizations are interested in multicloud strategies. Some organizations use more than one public cloud provider. It's easy to get into this. One business unit might use Microsoft technology so they pick Microsoft Azure. Another business unit is into machine learning so they use some Amazon AWS services.

Cloud providers offer different services. One cloud provider might be strong in one area and weak in another.



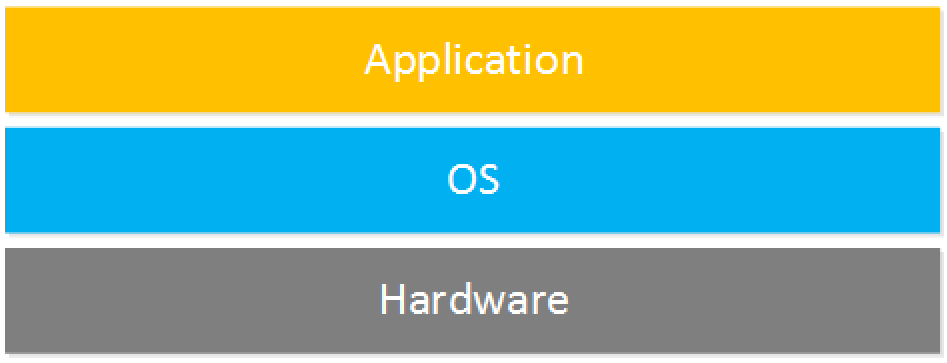
To connect to different public cloud providers you can use an **intercloud exchange**. The intercloud exchange offers you a connection to one or more public cloud providers.

Compute Virtualization (Containers and Virtual machines)

In this lesson, we'll take a look at the differences between virtual machines and containers.

Virtual Machines

Back in the days, a physical server would only run a single operating system with one or a few applications. We started with CPUs with a single core, then hyperthreading, and then multiple cores. The amount of memory in a single physical server also increased and became more affordable.

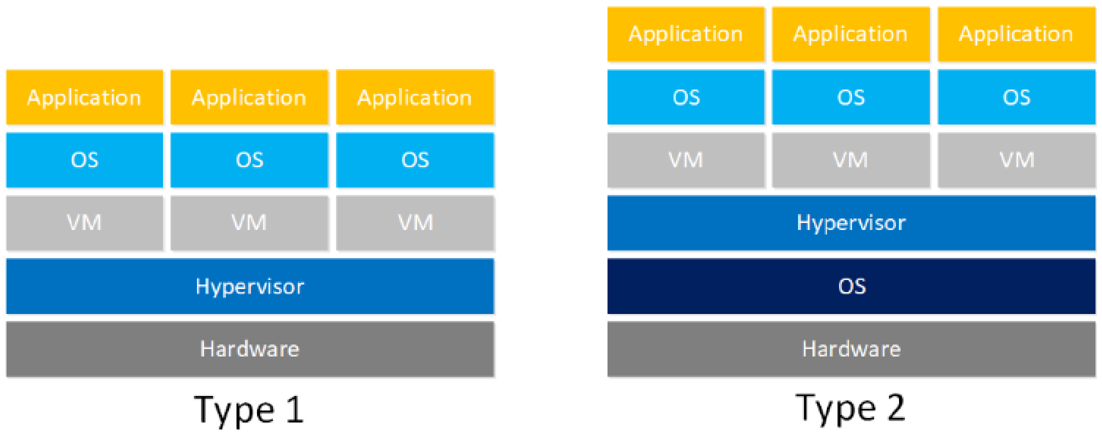


One of the reasons virtualizations is now so popular is that servers were under-utilized. We can easily run multiple operating systems and multiple applications on a single physical server. A virtual machine is pretty identical to a physical server except it's virtual. We have virtual hardware (CPUs, memory, storage, etc.) which runs on a hypervisor.

The hypervisor is the server virtualization software that runs on the physical server. This is where we create virtual machines and configure how much CPU cores, memory, storage, etc. each virtual machine has. There are two hypervisor types:

- **Type 1:** this hypervisor runs directly on hardware, which means you can assign more resources to virtual machines. Examples are VMware ESXi, Citrix Xen, KVM, and Microsoft Hyper-V.

- **Type 2:** this hypervisor runs on top of an operating system like Microsoft Windows, Apple MacOS, or Linux. We usually use a type 2 hypervisor on desktops or laptops. Two popular hypervisors are Oracle VM Virtualbox and VMWare Workstation. If you have never played with virtual machines before, give **Virtualbox** a try. You can download it for **free**.



Note:
Even a type 1 hypervisor has something of an OS. For example, VMWare ESXi uses a proprietary kernel called the VMkernel. It's very lightweight compared to a "full" operating system like Microsoft Windows or any Linux distribution.

When the hypervisor or the underlying physical server crashes, all your virtual machines disappear. Fortunately, there are products so you can migrate virtual machines from one physical server to another with no downtime or noticeable latency.

One advantage of virtual machines is that we are familiar with physical servers. It's easy to understand, it's a server, but virtual. We can use all the management and security tools we know to manage our physical or virtual servers.

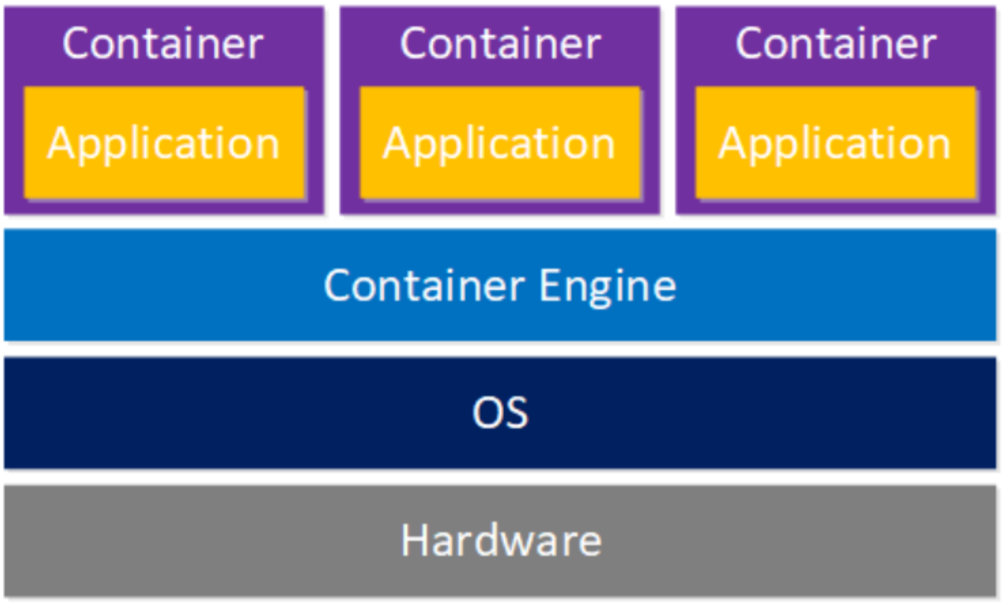
One disadvantage of virtual machines is that there is a lot of overhead. For example, let's say I want to run the freeradius application. A virtual machine **with** the Ubuntu 18.04 LTS operating system and nothing else installed is already ~3400 MB. Freeradius and its dependencies require about 5 MB. This brings my total storage to **3400 + 5 = 3405 MB**.

That's 3405 MB of storage only to run a simple application. Starting a virtual machine can take minutes, it has to boot the OS and start your applications.

Containers

Containers are sometimes called light-weight virtual machines, but they are nothing like virtual machines.

A container is a "package" that only contains an application and its dependencies, nothing more. This package is stored as a container image. Containers run on top of a container engine, which runs on top of an operating system. Starting a container is very quick since the operating system is already running. Containers are isolated from each other by the container engine.



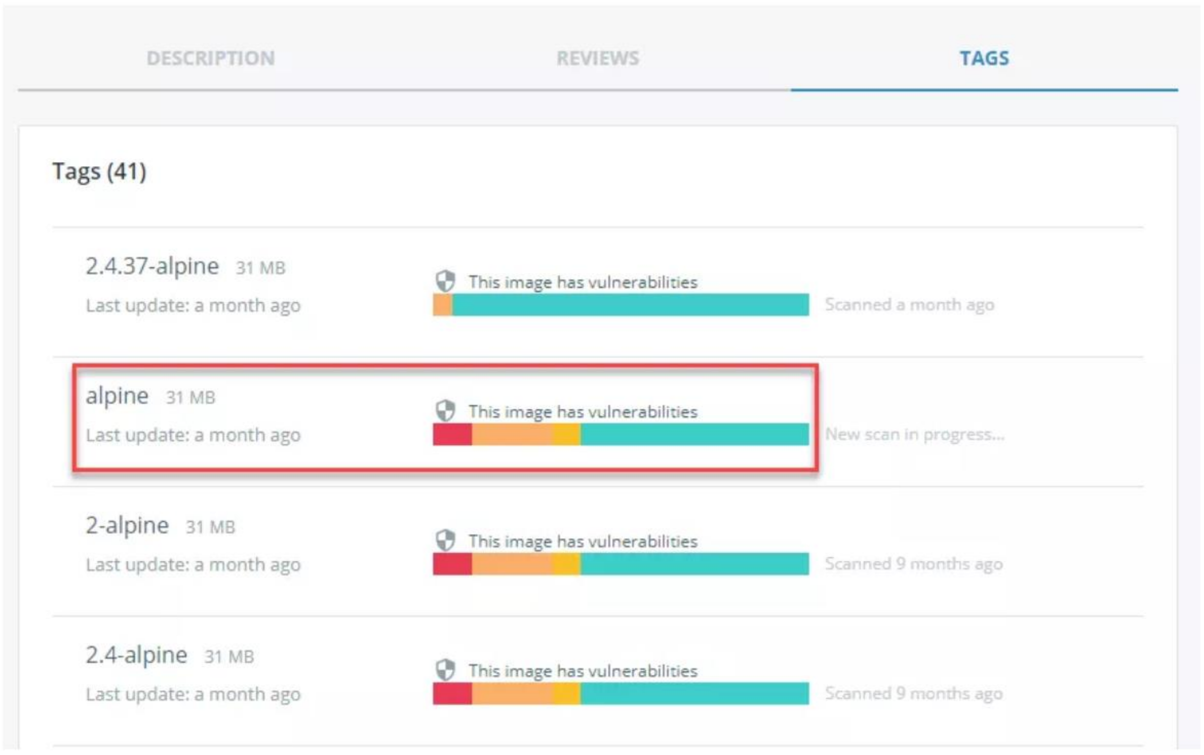
Containerization has many advantages:

1. **Small**: the container image only has the application and its dependencies, nothing more. This results in a small container image. For example, freeradius. It's only ~ 5 MB and has everything you need to run freeradius. https://gns3.com/marketplace/appliances/aaa-2
2. **Fast**: you don't have to start a virtual server with a virtual BIOS and virtual operating system. Spinning up a container is as fast as starting an application and only takes milliseconds.
3. **Portability**: The container image has everything the application needs. I can create a container image on my local machine, then ship somewhere else like a remote server. If it runs on my computer, it will run on other computers or servers.
4. **Isolation**: containers run on the same operating system but are isolated from each other. When one container crashes, it doesn't affect the other containers.
5. **Scalability**: You can add more identical containers when you need to scale out. Since containers are small and start quickly, you can scale out and in easily.
6. **Immutability**: container images are built based on a "**blueprint**". The freeradius image mentioned earlier is a Docker container; the blueprint is called a **Dockerfile**. When you change your code, you build a **new container image**.

Are there any disadvantages to containers? Containers are isolated from each other on the process level which could be less secure than a virtual machine which is fully isolated.

Another issue is security. Containers are based on a **blueprint** which is basically like a snapshot. You build the container image from the blueprint, and the container doesn't change anymore. If you want to update your container, you rebuild a new container image. This is different from a (virtual) server which you configure to install the latest security updates automatically.

Be careful that you run no container images that are outdated and might have vulnerabilities. For example, take a look at the Apache container images on Docker hub:


DESCRIPTION REVIEWS TAGS

Tags (41)

2.4.37-alpine 31 MB
Last update: a month ago
This image has vulnerabilities
Scanned a month ago

alpine 31 MB
Last update: a month ago
This image has vulnerabilities
New scan in progress...

2-alpine 31 MB
Last update: a month ago
This image has vulnerabilities
Scanned 9 months ago

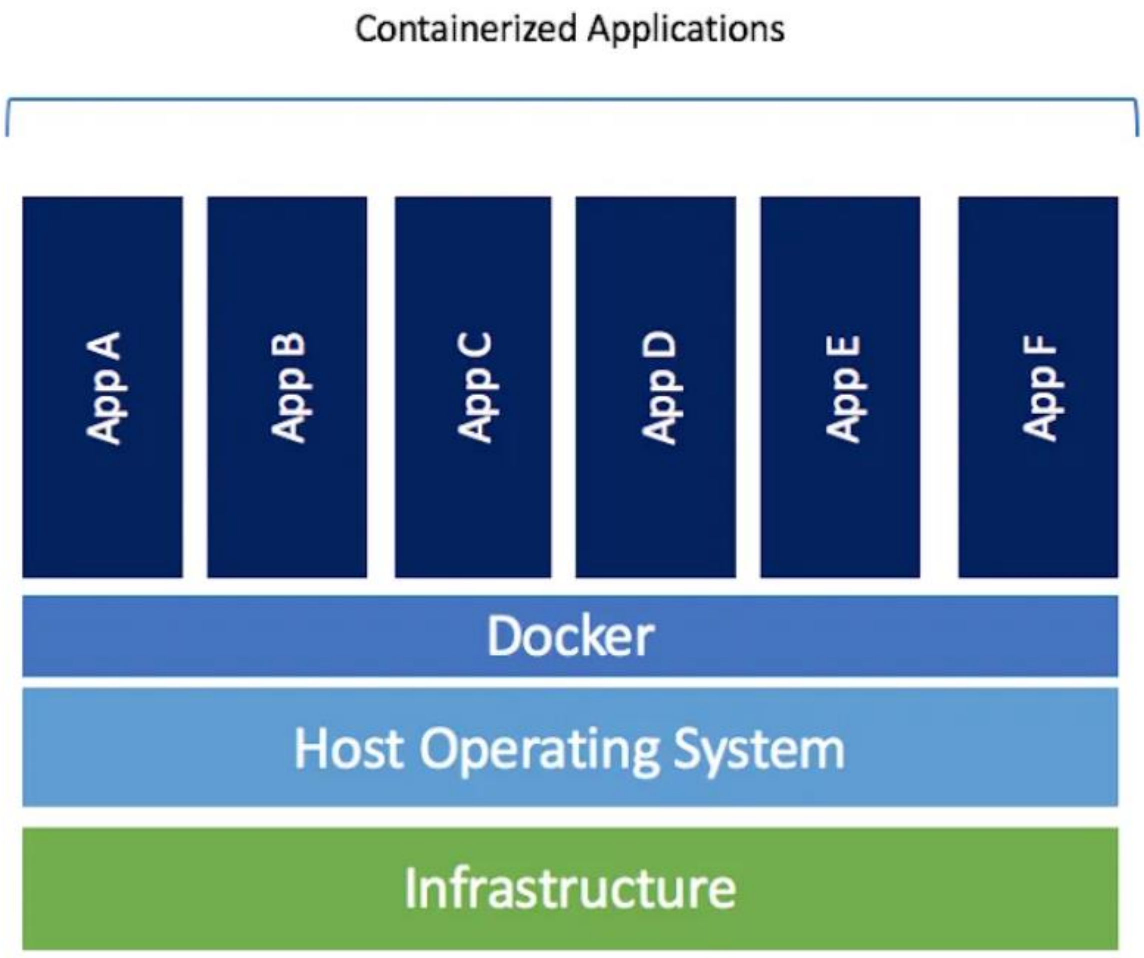
2.4-alpine 31 MB
Last update: a month ago
This image has vulnerabilities
Scanned 9 months ago

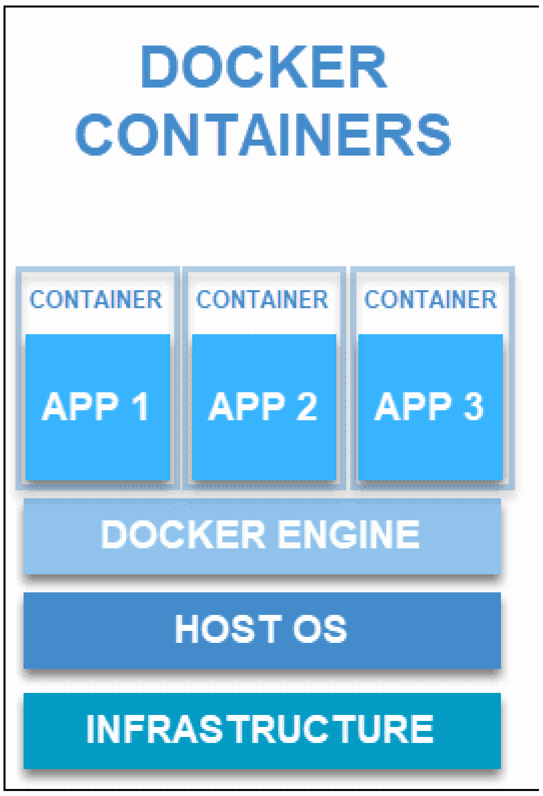
This vulnerability could be dangerous. It might also be unwise to run a public container image on your production network. When you build your own container images, you need to ensure you check them for security vulnerabilities and rebuild them when needed.

Docker is the most popular container platform. Other options:

- rkt (rocket)
- LXD
- Linux VServer
- Windows Containers

The main difference between Docker and container is that a **Docker is a platform to build, run and manage software containers** while a container is a lightweight software that provides operating system virtualization to run applications and its dependencies in resource isolated processes.





Introduction to Cloud Computing

Cloud computing is an approach where everything is delivered as a **service by cloud providers or the IT department of your company.**

1. **Bare Metal Servers**

Before **server virtualization,** you would only see **bare metal servers**. A bare metal server is a physical server that runs a single operating system like Microsoft Windows Server or Linux. This is from a time where CPUs only had 1-2 cores and not a lot a RAM.

Servers are similar to your computer at home, the main difference is that they are built to run 24/7 and usually have more reliable hardware. Some have redundant power supplies, hard disks, etc.

They come in different forms and shapes, the tower servers are often used in office buildings:

The bare metal server has a single operating system and can be used for one or multiple applications. For example, for small businesses, Microsoft Small Business Server has always been a popular office solution. It runs on 1 or 2 physical servers and offers:

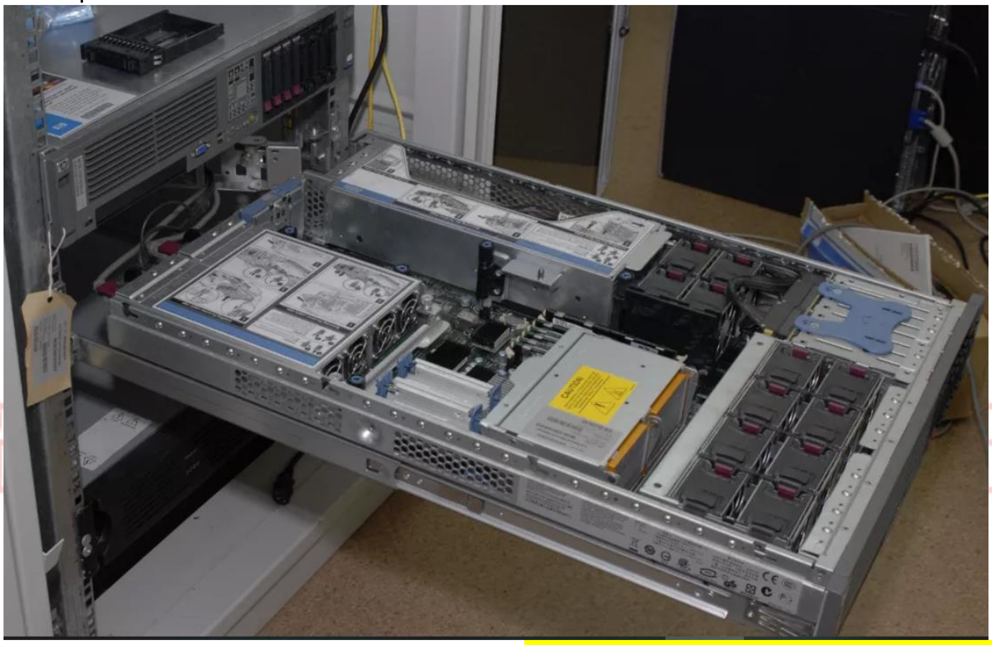
- Directory service
- DNS server
- E-mail server
- Web server
- File server
- And some other services

In enterprise environments, it’s more common to see that a physical server is used for each “role”. One web server, one DNS server, etc.

In data centers, physical space is expensive so it’s more common to see rack servers there:

Rack servers are placed in server cabinets:

Here's an example of an HP Proliant rack server in a rack cabinet:



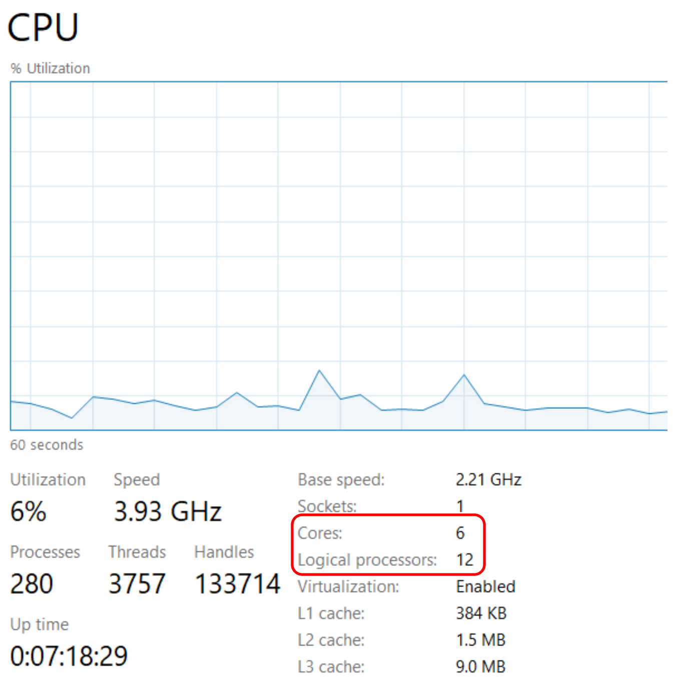
Later, blade servers were introduced that offer even more computing power / memory and use less space:

Above you see a blade enclosure that fits in a server rack. It offers networking, power, and cooling. The *blade servers* have CPUs, memory, and storage. They fit in the slots of the blade enclosure.

When you buy a physical server, you have to think of the required resources. How much memory does the server need? How much disk space? How fast should the CPUs be? etc. You also have to take future growth into account.

2. Server Virtualization

In the last decade, the number of cores in a single processor has grown rapidly. We went from single core processors to dual core processors, quad core processors and now there are even processors with more than 10-20 cores. There is a also a technique called *hyperthreading* where we have two virtual cores for each physical core. This allows the operating system to execute two threads at the same time. Here is an example of an Intel I7 processor:



```
Utilization   Speed          Base speed:          2.21 GHz
6%            3.93 GHz       Sockets:             1
                             Cores:               6
Processes  Threads  Handles  Logical processors:  12
280        3757     133714   Virtualization:      Enabled
                             L1 cache:            384 KB
Up time                      L2 cache:            1.5 MB
0:07:18:29                   L3 cache:            9.0 MB
```

Above you can see that this CPU has 6 physical cores and 12 virtual (logical) cores.

The amount of (affordable) RAM has also increased a lot. A single physical server often has way more CPU and RAM resources than what is required for a single operating system. Nowadays, we use a lot of server virtualization which means that we run **multiple virtual machines on a single physical server**.

Here are some companies/products that offer server virtualization solutions:

- VMware
- Microsoft HyperV
- Citrix Xenserver
- Red Hat KVM

3. Cloud Computing

The idea behind cloud computing is that we offer different services, a customer should be able to request the service and receive the service right away. There is no human in between that has to look at the customer's request, process it and report back. Everything is automated.

That's not the only advantage of cloud computing, though. NIST (National Institute of Standards and Technology has a good definition of cloud computing:

- **On-demand self-service:** a customer should be able to get a service automatically, without having to wait for a human to provide it for her. The customer should also be able to terminate the service by herself.

- **Broad network access:** the service should be available using a variety of different platforms including computers, tablets, smartphones. We should also be able to reach the service using different connections, including the Internet or private WAN connections.
- **Resource pooling:** the cloud provider should not assign fixed resources to the service but it has to be dynamic. For example, when a website suddenly receives a lot of traffic, multiple web servers should be created automatically so that we can deal with the incoming traffic.
- **Rapid elasticity:** to the customer, the resources should look to be unlimited. For example, there are some cloud backup providers that you can use for remote backups. You pay for the service and behind the scenes, they will take care that there is enough storage space for you to upload your files to. It doesn't matter if you upload 1 GB or 100 TB.
- **Measured service:** the cloud provider measures all resource usage for billing and transparency.

CISCO SWITCH VIRTUALIZATION

1. Virtual Network Function (VNF)

Nowadays, we can also use virtual solutions. A **virtual network function (VNF)** is the virtual version of a hardware device's network function. VNFs are available as virtual machines or containers.

Here are examples of Cisco VNFs:

- vEdge Cloud
- CSR1000v
- ASAV
- Cloud Services Platform (CSP) 2100
- XRv 9000
- Firepower NGFWv
- Web Security Virtual Appliance (WSAv)
- Email Security Virtual Appliance (ESAv)
- Advanced Malware Protection Virtual (AMPv)

The advantages of VNFs are similar to server virtualization and cloud computing. We have a **shorter time to market (TTM)** because, without the hardware requirement, we can quickly launch a new network function as a virtual machine or container.

2. Cisco Stackwise

Cisco's access layer switches used to be all separate physical switches where we use Ethernet cables for connectivity between the switches. Cisco Stackwise changed this, it allows us to turn multiple physical switches **into a single logical switch**.

Switches that support Stackwise use a special stacking cable to connect the switches to each other. Each switch has two stacking connectors that are used to "daisy-chain" (loop) the switches together.

Each switch is connected to the one below it and the bottom switch will be connected to the one on top.

The Stackwise cable is like an extension of the switching fabric of the switches. When an Ethernet frame has to be moved from one physical switch to another, the Stackwise "loop" is used. The advantage of using a cabled loop is that you can remove one switch from the stack, the loop will be broken but the stack will keep working.

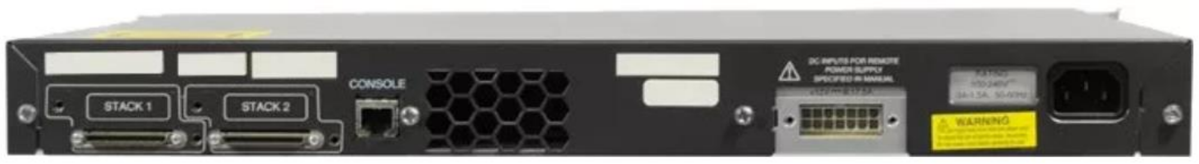
One switch in the stack becomes the **master** that does all "management tasks" for the stack. All other switches are **members**. If the master fails, another member will become the new master. To select a master, Stackwise uses an election process that checks for the following criteria (in order of importance):

1. **User priority**: we can configure a priority to decide which switch becomes the master.
2. **Default Configuration**: A switch that already has a configuration will take precedence over switches with no configuration.
3. **Hardware/software priority**: The switch with the most extensive feature set has a higher priority than another switch (for example: IP Services vs IP base).
4. **Uptime**: The switch with the longest uptime.
5. **MAC address**: The switch with the lowest MAC address.

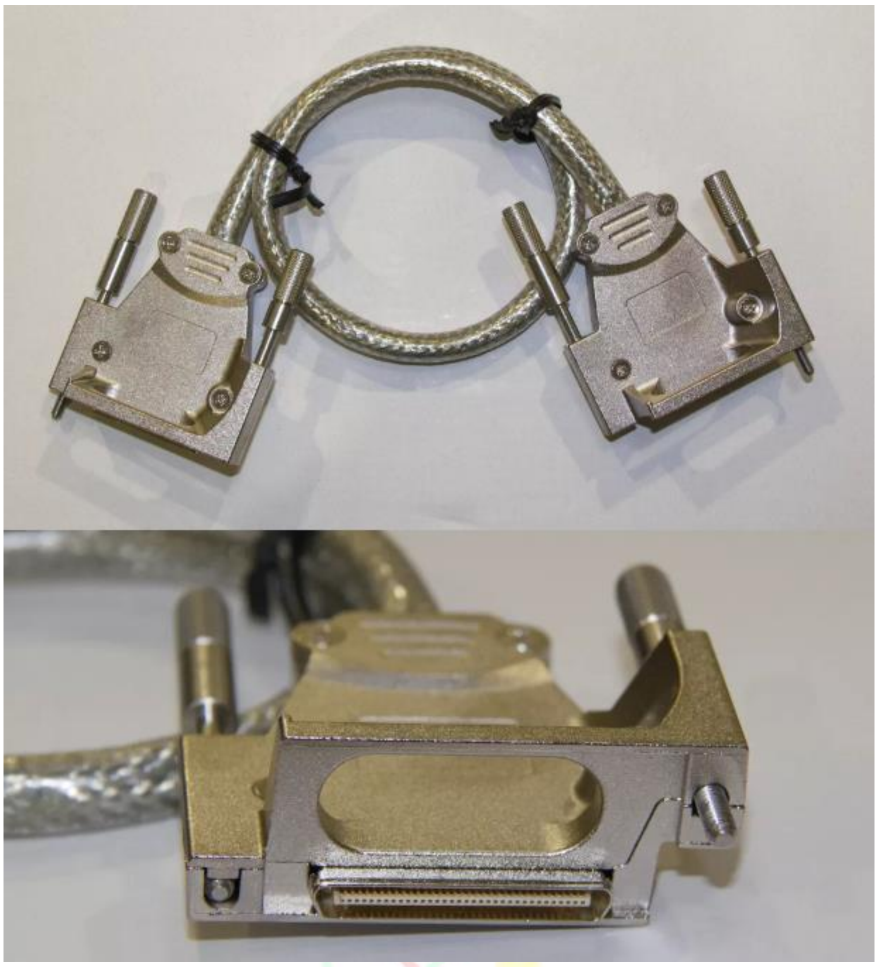
It makes sense to choose the master ourselves so normally we use user priority to configure the master.

Once the stack has been created, the configuration of the switches is the same as if it were one single switch…they share the same management IP address, hostname, etc.

Here's a picture of the stacking connectors, this is the rear of a Cisco Catalyst 3750 switch:



You can see the two connectors on the left side…Stack1 and Stack 2. Here's what the Stackwise cable looks like:

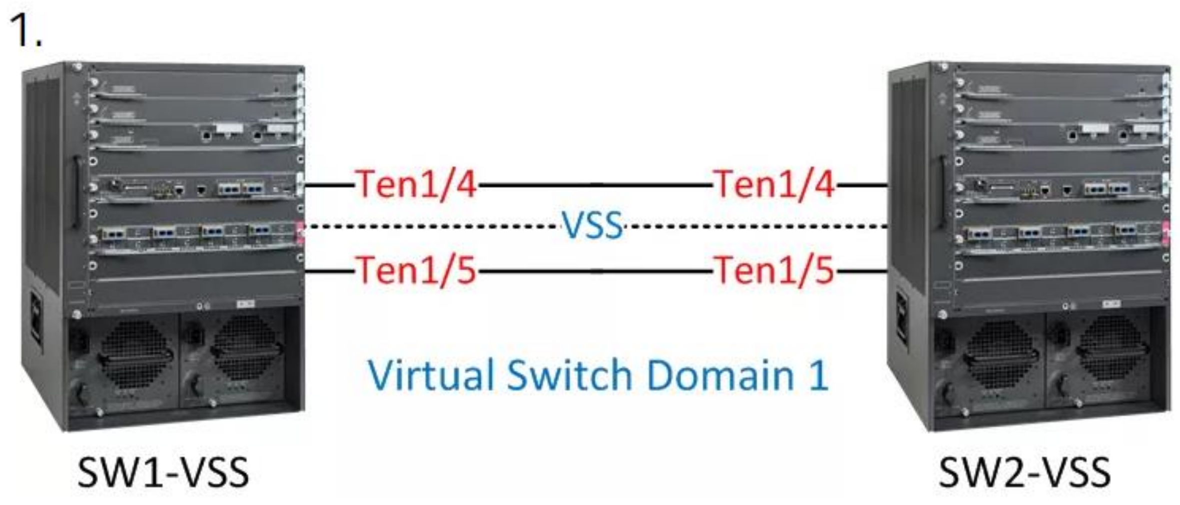


3. Cisco 6500 VSS

The Virtual Switching System (VSS) allows two Cisco Catalyst 6500 or 4500 chassis to bond together so that is seen as a single virtual swich to the rest of the network. Other devices will see the VSS configured 6500 as a single device which means it's possible to use multi chassis etherchannel and protocols like spanning-tree will only see a single switch.

Some other features are NSF (Non Stop Forwarding) / SSO (Stateful Switchover) which means that when a single chassis fails the other one will take over without any downtime since the routing table / CEF table etc. are stored in both chassis' supervisors.

Another cool feature is EFSU (Enhanced Fast Software Upgrade) which allows you to upgrade the IOS version without any downtime.



Right now I have two 6500s that are running in “standalone”. In order to bond these two using VSS we will have to do the following:

- Configure a **virtual switch domain** on both switches and configure one switch as “switch 1” and the other one as “switch 2”.
- Configure the **virtual switch links**.
- Execute the **conversion** command which will reboot the switches.