Cloud Computing Virtualization

Virtualization is a technique, which allows to share single physical instance of an application or resource among multiple organizations or tenants (customers). It does so by **assigning a logical name** to a physical resource and providing a **pointer to that physical resource** on demand.

Virtualization Concept

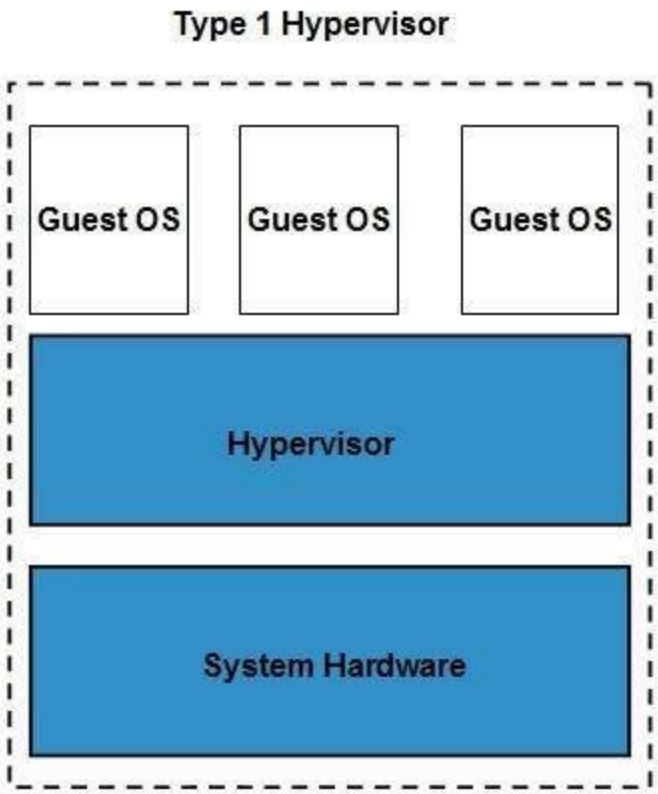
Creating a virtual machine over existing operating system and hardware is referred as Hardware Virtualization. Virtual Machines provide an environment that is logically separated from the underlying hardware.

The machine on which the virtual machine is created is known as **host machine** and **virtual machine** is referred as a **guest machine.** This virtual machine is managed by a software or firmware, which is known as **hypervisor.**

Hypervisor

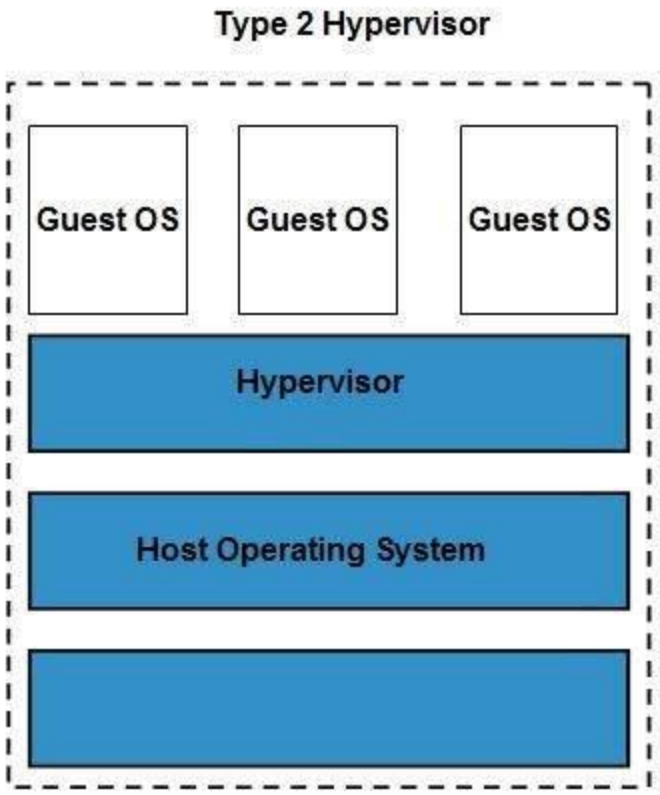
The **hypervisor** is a firmware or low-level program that acts as a Virtual Machine Manager. There are two types of hypervisor:

Type 1 hypervisor executes on bare system. LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogic VLX are examples of Type 1 hypervisor. The following diagram shows the Type 1 hypervisor.



The **type1 hypervisor** does not have any host operating system because they are installed on a bare system.

Type 2 hypervisor is a software interface that emulates the devices with which a system normally interacts. Containers, KVM, Microsoft Hyper V, VMWare Fusion, Virtual Server 2005 R2, Windows Virtual PC and **VMWare workstation 6.0** are examples of Type 2 hypervisor. The following diagram shows the Type 2 hypervisor.



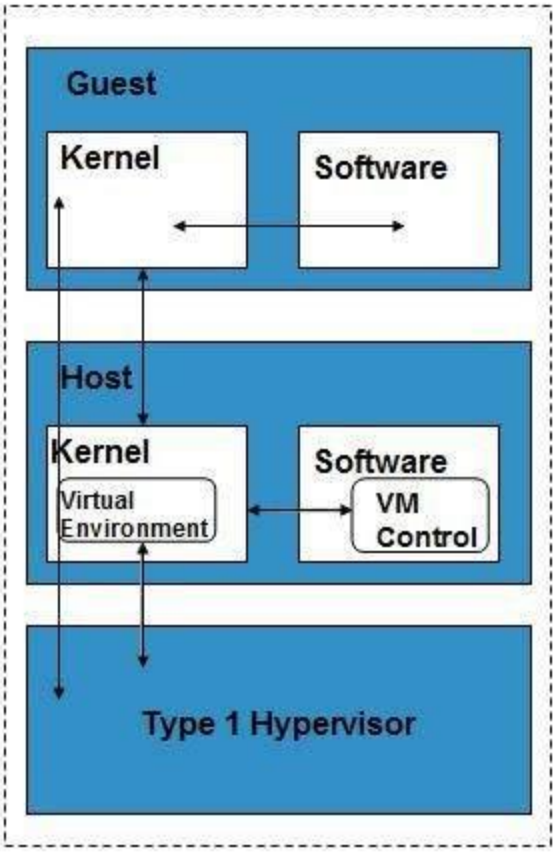
Types of Hardware Virtualization

Here are the three types of hardware virtualization:

- Full Virtualization
- Emulation Virtualization
- Paravirtualization

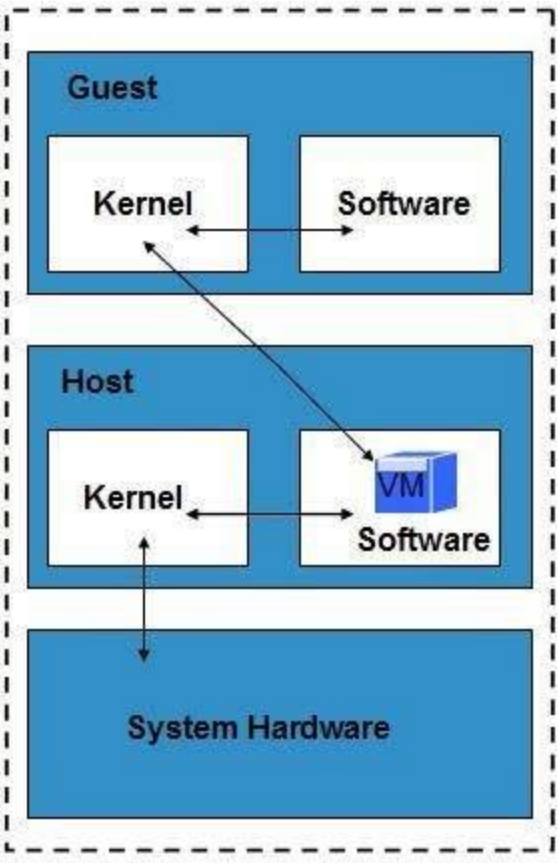
Full Virtualization

In **full virtualization,** the underlying hardware is completely simulated. Guest software does not require any modification to run.



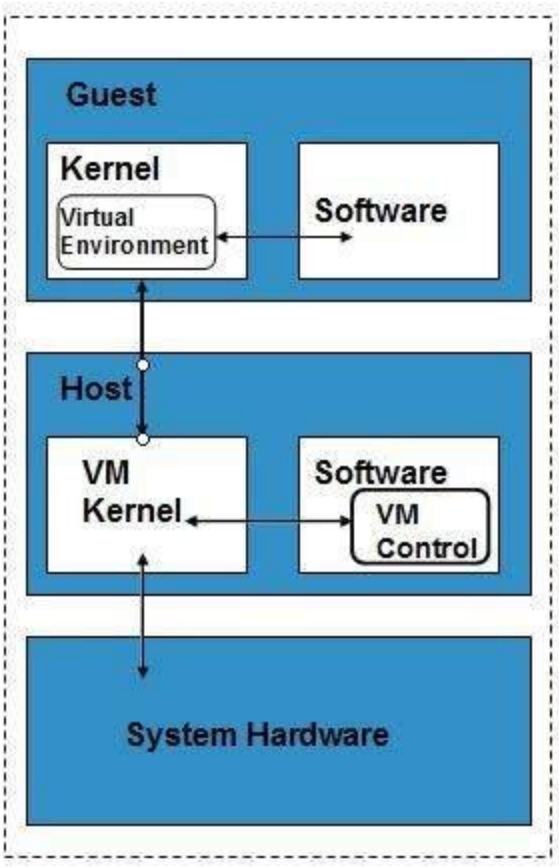
Emulation Virtualization

In **Emulation,** the virtual machine simulates the hardware and hence becomes independent of it. In this, the guest operating system does not require modification.



Paravirtualization

In **Paravirtualization,** the hardware is not simulated. The guest software run their own isolated domains.



VMware vSphere is highly developed infrastructure that offers a management infrastructure framework for virtualization. It virtualizes the system, storage and networking hardware.

Tools for Virtualization

Simply put, virtualization is the process of creating a virtual instance of a technology resource like storage, memory, or an operating system; instead of deploying the actual resource. Essentially, virtualization software can convert a single computer into multiple ones. Virtualization has two dimensions: interaction between a virtualized operating system and the underlying hardware and interaction between a virtual host and physical host.

Virtualization software operates the same way as any other software application. Once the virtualization software has been loaded, an operating system can be installed from an .iso file or installation disc. The installed OS now runs on a virtual environment referred to as the virtual machine. The main OS becomes the ‘host’ while any other additional OS becomes the ‘guest’. Additional OS that is installed acts like an entirely new machine.

Virtualization programs that manage the creation and operating of virtual machines are referred to as hypervisors. Some virtualization products may also use the CPU technology virtualization extensions to increase the performance of VMs. This is referred to as hardware-assisted virtualization. The main advantage of virtualization is the reduced cost, since there is less hardware to be deployed. Others include: centralized management of several resources on one machine and flexibility in the creation of a hardware-agnostic computing environment.

Ganeti

Ganeti is a cluster server management tool developed by Google; it is built on existing virtualization technologies like KVM, Xen and other Open Source software. Ganeti was initially started as a VMware alternative for managing networks, storage, and virtual machines- and not as a cloud platform, meaning it lacks several of the features that come with larger open cloud projects. It was designed to handle cluster management of virtual servers and offer quick and easy recovery after physical failures using commodity software.

For Ganeti to work, virtualization software must be pre-installed on the servers. Once installed, it assumes management of the virtual instances; Ganeti handles startup, shutdown, OS installation, migration, disk creation, and can be used to preemptively relocate a VM off a physical machine that is failing. Ganeti can be used for both single-host management, like Xen Tools or Libyirt, and large-scale computing on an OpenStack level.

KVM (Kernel-based Virtual Machine)

KVM is an open source virtualization tool for Linux and contains virtualization extensions (AMD-V or Intel VT). It can either be operated in emulation or hardware mode; however, without the CPU extensions, the overall performance will be poor. Despite the fact that it was designed for command line, KVM has a decent management interface that enable users to perform actions like launching and stopping virtual machines or taking screen shots with ease.

The interface (Virtual Machine Manger – VMM) can also be used to control Xen virtual machines. Each of virtual machine within KVM has personalized virtualized hardware: a graphics adapter, disk, network adapter, etc. If you are searching for a free modern virtualization solution with an unlimited usage mode and without additional feature tearing or licensing fees, and a powerful command line interface then KVM is your best bet.

oVirt

oVirt is a virtualization solution used to manage/create virtual datacenters. oVirt manages storage options, virtualized networks, and virtual machines using interactive an easy to use web-based administration and user portal. oVirt supports several advanced virtualization features like live storage migration, high availability, and the ability to control and schedule the deployment of virtual machines.

Aside from the oVirt engine, this Red Hat backed project also includes other components, like oVirt Node – a scaled down version of Linux with enough code to host virtual machines. The oVirt project also includes data reporting and warehouse components, based on the work of open source software providers Jaspersoft and Talend.

Packer

Packer can be used by system admins to build then subsequently manage the operations of virtual machine images. The same commands and files can be used to build an image on Digital Ocean, AWS or for vagrant and VirtualBox. This enables you to use the same system for development which you then create in production.

Packer is notably light, high performing, and operates on every major operating system. It assembles and configures all the necessary components for a virtual machine then creates images that run on multiple platforms. Packer doesn't replace configuration management tools like Puppet or Chef; as a matter of fact, when creating images, Packer can utilize tools like Puppet or Chef to install applications onto the image.

Vagrant

Vagrant is a command-line tool that provides a framework and configuration format for creating, managing and distributing virtualized development environments. These environments can live on your computer or in the cloud, and are portable between Linux, Mac OS X, and Windows.

Vagrant has a differentiating feature – Vagrant Share that enables users to share their running Vagrant environment via the internet. This makes it easy to collaborate and share on development environments thus creating consistent working environments for teams of software developers using a virtual machine. Vagrant can also work alongside configuration management tools like Puppet and Chef.

VirtualBox

VirtualBox is an open source cross-platform product that is currently under the stewardship of Oracle. VirtualBox is one of the veterans of the virtualization scene, and remains a lightweight and reliable virtualization tool that is easy to install and use.

With its latest release, VirtualBox adds support for touchscreens and other user-recognizable improvements like Webcam pass-thru, support for IPv6 with VRDP and, video capture support. One important feature VirtualBox lacks is the ability to boot from existing Boot Camp partitions.

Xen

Xen is a hypervisor that started out as a Microsoft backed startup at the University of Cambridge and has now risen to become one of the best Linux hypervisors. The Xen Hypervisor is inserted between the server's hardware and the operating system. This creates an abstraction layer that allows multiple guest operating systems to be concurrently executed on a single physical server. Xen is included with most popular Linux distributions like Fedora, RHEL, CentOS, Ubuntu, and Debian.

Xen supports 'hardware assisted' virtualization and para-virtualization for unmodified and modified guests respectively. The guests can be windows or Linux, but most guests particularly in the hosting space are Linux variants.

Virtual Machine Monitor (VMM)

Definition

A Virtual Machine Monitor (VMM) is a software program that enables the creation, management and governance of virtual machines (VM) and manages the operation of a virtualized environment on top of a physical host machine.

VMM is also known as Virtual Machine Manager and Hypervisor. However, the provided architectural implementation and services differ by vendor product.

VMM is the primary software behind virtualization environments and implementations. When installed over a host machine, VMM facilitates the creation of VMs, each with separate operating systems (OS) and applications. VMM manages the backend operation of these VMs by allocating the necessary computing, memory, storage and other input/output (I/O) resources.

VMM also provides a centralized interface for managing the entire operation, status and availability of VMs that are installed over a single host or spread across different and interconnected hosts.

VIRTUAL MACHINE MONITOR:

A Virtual Machine Monitor (VMM) is a software program that enables the creation, management and governance of virtual machines (VM) and manages the operation of a virtualized environment on top of a physical host machine. VMM is also known as Virtual Machine Manager and Hypervisor. However, the provided architectural implementation and services differ by vendor product. VMM is the primary software behind virtualization environments and implementations. When installed over a host machine, VMM facilitates the creation of VMs, each with separate operating systems (OS) and applications. VMM manages the backend operation of these VMs by allocating the necessary computing, memory, storage and other input/output (I/O) resources. In a virtual machine environment, the virtual machine monitor (VMM) becomes the master control program with the highest privilege level, and the VMM manages one or more operating systems, now referred to as "guest operating systems." Each

guest OS manages its own applications as it normally does in a non-virtual environment, except that it has been isolated in the computer by the VMM. Each guest OS with its applications is known as a "virtual machine" and is sometimes called a "guest OS stack." VMM Types Following are the three common VMM architectures, showing the relationship between the VMM, the guest OS and device drivers. All three methods can be paravirtualized (changes in the guest OS are made) or fully virtualized (no changes in the guest OS). The term "hypervisor" is used to refer to the virtual machine monitor component nearest the hardware. 1. Host OS – This method helps in enabling VMM in order to get installed on guest OS and running computer with no modification. 2. Hypervisor – It offers best performance, flexibility, and most control in VM (Virtual machine) environment. 3. Service OS – It combines hypervisor's robustness with hosted model's flexibility that uses existing OS. Implementing Virtual Machine Monitors Virtual machine monitors can be implemented in two ways. First, one can run the monitor directly on hardware in kernel mode, with the guest operating systems in user mode. Second, one can run the monitor as an application in user mode on top of a host operating system. The latter may be less complex to implement because the monitor can take advantage of the abstractions provided by the host operating systems, but it is only possible if the host operating system forwards all the events that monitor needs to perform its job. Virtual machine properties A virtual machine (VM) is implemented by adding a layer of software to a real machine to support the desired virtual machine's architecture. For example, virtualizing software installed on an Apple Macintosh can provide a Windows/IA-32 virtual machine capable of running PC application programs. In general, a virtual machine can circumvent real machine compatibility constraints and hardware resource constraints to enable a higher degree of software portability and flexibility. A wide variety of virtual machines exist to provide an equally wide variety of benefits. Multiple, replicated virtual machines can be implemented on a single hardware platform to provide individuals or user groups with their own operating system environments. The different system environments (possibly with different operating systems)also provide isolation and enhanced security. A large multiprocessor server can be divided into smaller virtual servers, while retaining the ability to balance the use of hardware resources across the system. Virtual machines can also employ emulation techniques to support cross platform software compatibility. For example, a platform implementing the PowerPC instruction set can be converted into a virtual platform running the IA-32 instruction set. Consequently, software written for one platform will

run on the other. This compatibility can be provided either at the system level (e.g.,to run a Windows OS on a Macintosh)or at the program or process level (e.g.,to run Excel on a Sun Solaris/SPARC platform).In addition to emulation, virtual machines can provide dynamic, on-the-fly optimization of program binaries. Finally, through emulation, virtual machines can enable new, proprietary instruction sets,e.g.,incorporating very long instruction words(VLIWs), while supporting programs in an existing, standard instruction set. The virtual machine examples just described are constructed to match the architectures of existing real machines. However, there are also virtual machines for which there are no corresponding real machines. It has become common for language developers to invent a virtual machine tailored to a new high-level language. Programs written in the high-level language are compiled to “binaries” targeted at the virtual machine. Then any real machine on which the virtual machine is implemented can run the compiled code. The power of this approach has been clearly demonstrated with the Java high level language and the Java virtual machine, where a high degree of platform independence has been achieved, thereby enabling a very flexible network computing environment. Virtual machines have been investigated and built by operating system developers, language designers, compiler developers, and hardware designers. Although each application of virtual machines has its unique characteristics, there also are underlying concepts and technologies that are common across the spectrum of virtual machines. Because the various virtual machine architectures and underlying technologies have been developed by different groups, it is especially important to unify this body of knowledge and understand the base technologies that cut across the various forms of virtual machines. The goals of this book are to describe the family of virtual machines in a unified way, to discuss the common underlying technologies that support them, and to demonstrate their versatility by exploring their many applications. HLLVM(High Level Language Virtual Machine) For the process VMs cross-platform portability is clearly a very important objective. For example, the FX!32 system enabled portability of application software compiled for a popular platform (IA-32 PC) to a less popular platform (Alpha). However, this approach allows cross-platform compatibility only on a case-by-case basis and requires a great deal of programming effort. For example, if one wanted to run IA-32 binaries on a number of hardware platforms currently in use, e.g., SPARC, PowerPC, and MIPS, then an FX!32-like VM would have to be developed for each of them. Full cross-platform portability is more easily achieved by taking a step back and designing it into an overall software framework. These

highlevel language VMs (HLL VMs) are similar to the process VMs described earlier. However, they are focused on minimizing hardware-specific and OS-specific features because these would compromise platform independence. High-level language VMs first became popular with the Pascal programming environment. .In a conventional system, Figure 1.10a, the compiler consists of a frontend that performs lexical, syntax, and semantic analysis to generate simple intermediate code — similar to machine code but more abstract. Typically the intermediate code does not contain specific register assignments, for example. Then a code generator takes the intermediate code and generates a binary containing machine code for a specific ISA and OS. This binary file is distributed and executed on platforms that support the given ISA/OS combination. To execute the program on a different platform, however, it must be recompiled for that platform. In HLL VMs, this model is changed (Figure 1.10b). The steps are similar to the conventional ones, but the point at which program distribution takes place is at a higher level. As shown in Figure 1.10b, a conventional compiler frontend generates abstract machine code, which is very similar to an intermediate form. An advantage of an HLL VM is that software is easily portable, once the VM is implemented on a target platform. While the VM implementation would take some effort, it is a much simpler task than developing a compiler for each platform and recompiling every application when it is ported. It is also simpler than developing a conventional emulating process VM for a typical real-world ISA. The Sun Microsystems Java VM architecture (Lindholm and Yellin 1999) and the Microsoft common language infrastructure (CLI), which is the foundation of the .NET framework (Box 2002), are more recent, widely used examples of HLL VMs. HyperVisors Hypervisors are virtual machine monitor(VMM) that enables numerous virtual operating systems to simultaneously run on a computer system. These virtual machines are also referred as guest machines and they all share the hardware of the physical machine like memory, processor, storage and other related resources. This improves and enhances the utilization of the underlying resources. The hypervisor isolates the operating systems from the primary host machine. The job of a hypervisor is to cater to the needs of a guest operating system and to manage it efficiently. Each virtual machine is independent and do not interfere with each another although they run on the same host machine. They are no way connected to one another. Even at times one of the virtual machines crashes or faces any issues, the other machines continue to perform normally. Hypervisors are divided into two types Type -1: is the bare-metal hypervisor that are deployed directly over the host's system hardware without any underlying

operating systems or software. Some examples of the type 1 hypervisors are Microsoft Hyper-V hypervisor, VMware ESXi, Citrix XenServer. Type 2: is a hosted hypervisor that runs as a software layer within a physical operating system. The hypervisor runs as a separate second layer over the hardware while the operating system runs as a third layer. The hosted hypervisors include Parallels Desktop and VMware Player.

VIRTUALIZATION FOR CLOUD

Virtualization is a technology that helps us to install different Operating Systems on a hardware. They are completely separated and independent from each other. In Wikipedia, you can find the definition as – “In computing, virtualization is a broad term that refers to the abstraction of computer resources.

Virtualization hides the physical characteristics of computing resources from their users, their applications or end users. This includes making a single physical resource (such as a server, an operating system, an application or a storage device) appear to function as multiple virtual resources. It can also include making multiple physical resources (such as storage devices or servers) appear as a single virtual resource...”

Virtualization is often −

- The creation of many virtual resources from one physical resource.
- The creation of one virtual resource from one or more physical resource.

Types of Virtualization

Today the term virtualization is widely applied to a number of concepts, some of which are described below −

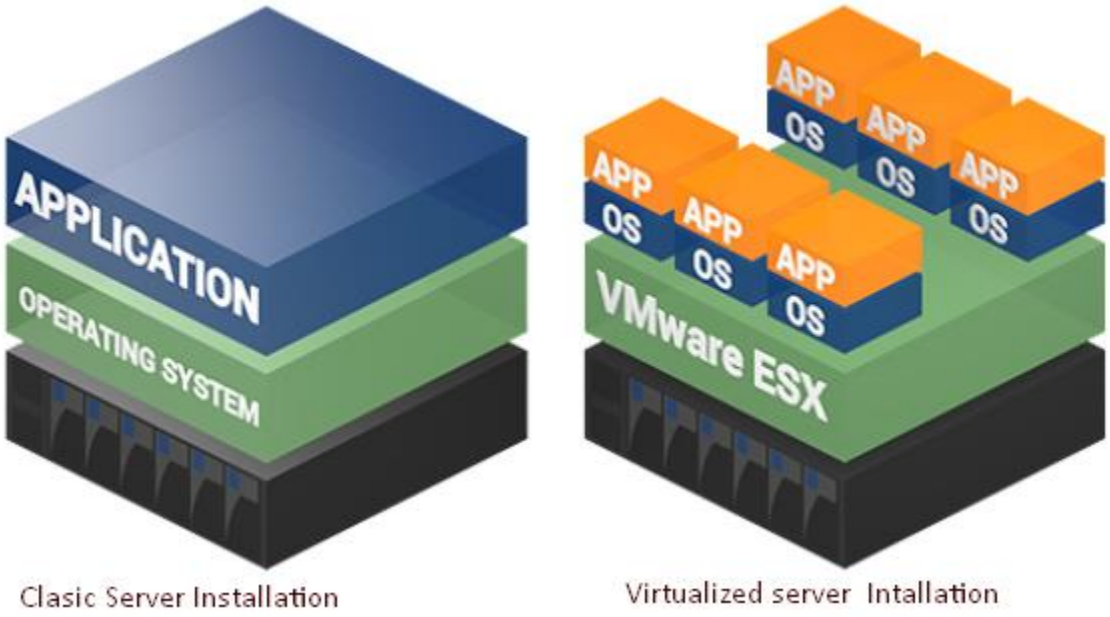
- Server Virtualization
- Client & Desktop Virtualization
- Services and Applications Virtualization

- Network Virtualization
- Storage Virtualization

Let us now discuss each of these in detail.

Server Virtualization

It is virtualizing your server infrastructure where you do not have to use any more physical servers for different purposes.

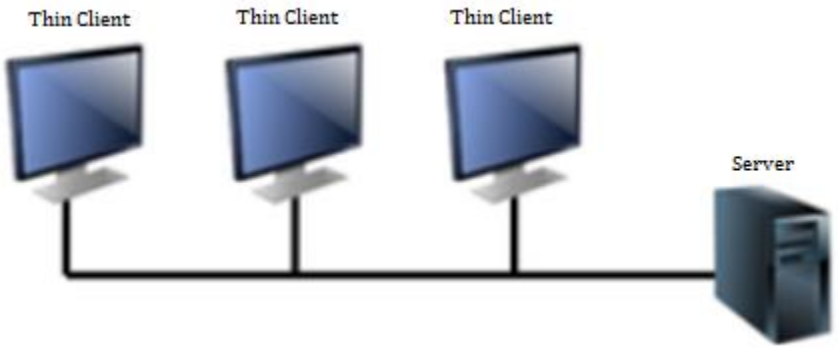


Clasic Server Installation

Virtualized server Intallation

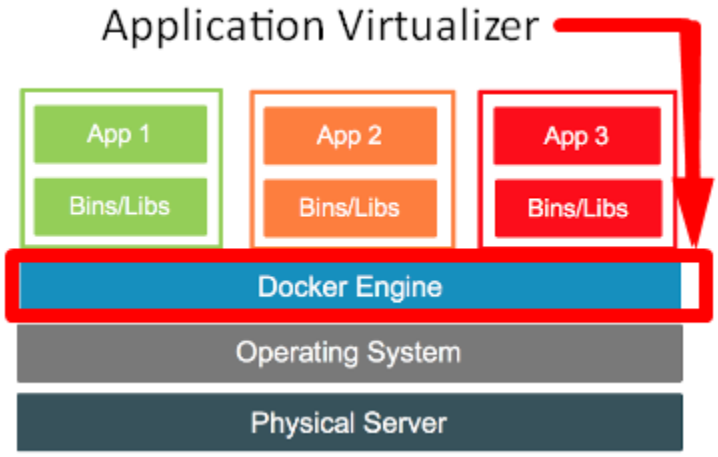
Client & Desktop Virtualization

This is similar to server virtualization, but this time is on the user's site where you virtualize their desktops. We change their desktops with thin clients and by utilizing the datacenter resources.



Services and Applications Virtualization

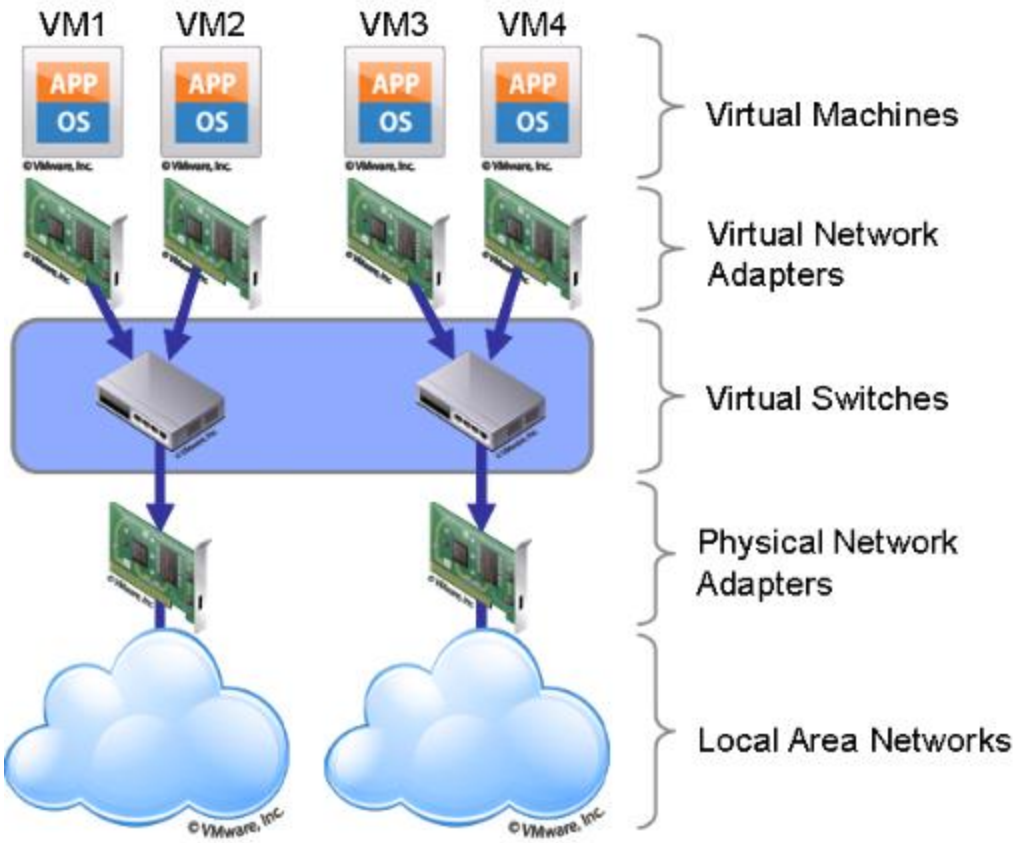
The virtualization technology isolates applications from the underlying operating system and from other applications, in order to increase compatibility and manageability. For example – Docker can be used for that purpose.



Network Virtualization

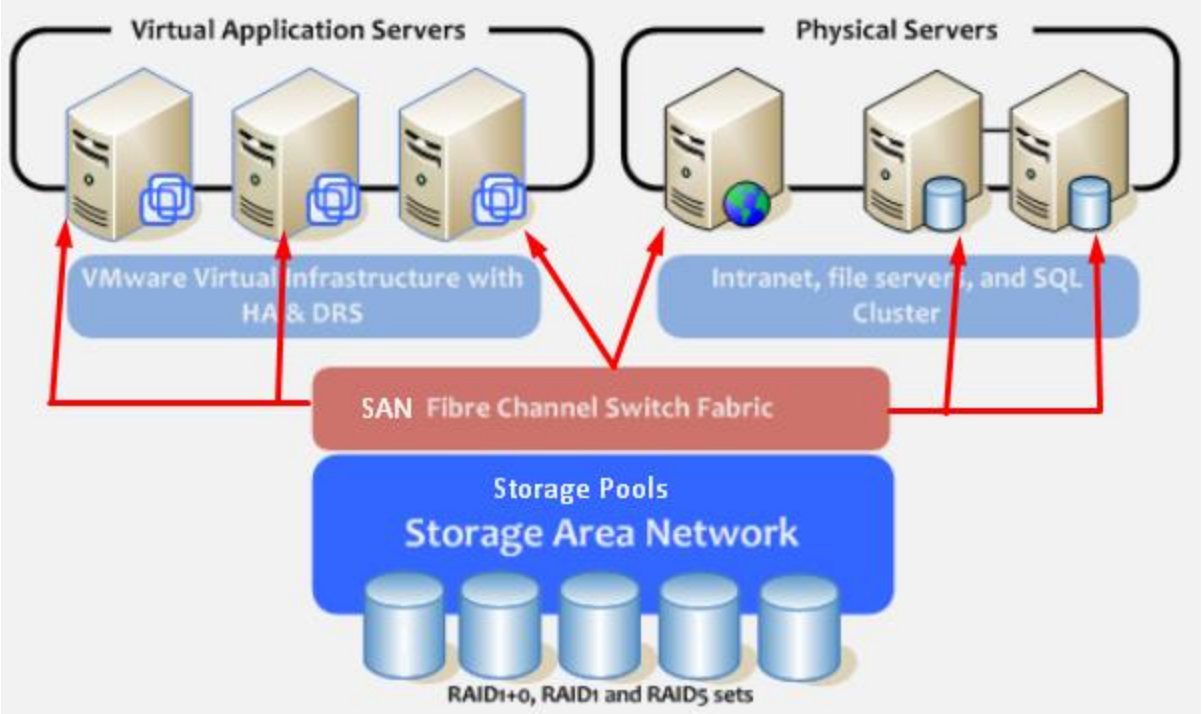
It is a part of virtualization infrastructure, which is used especially if you are going to visualize your servers. It helps you in creating multiple switching, Vlans, NAT-ing, etc.

The following illustration shows the VMware schema −



Storage Virtualization

This is widely used in datacenters where you have a big storage and it helps you to create, delete, allocated storage to different hardware. This allocation is done through network connection. The leader on storage is SAN. A schematic illustration is given below −



Understanding Different Types of Hypervisors

A hypervisor is a thin software layer that intercepts operating system calls to the hardware. It is also called as the **Virtual Machine Monitor** (VMM). It creates a virtual platform on the host computer, on top of which multiple guest operating systems are executed and monitored.

Hypervisors are two types −

- Native of Bare Metal Hypervisor and
- Hosted Hypervisor

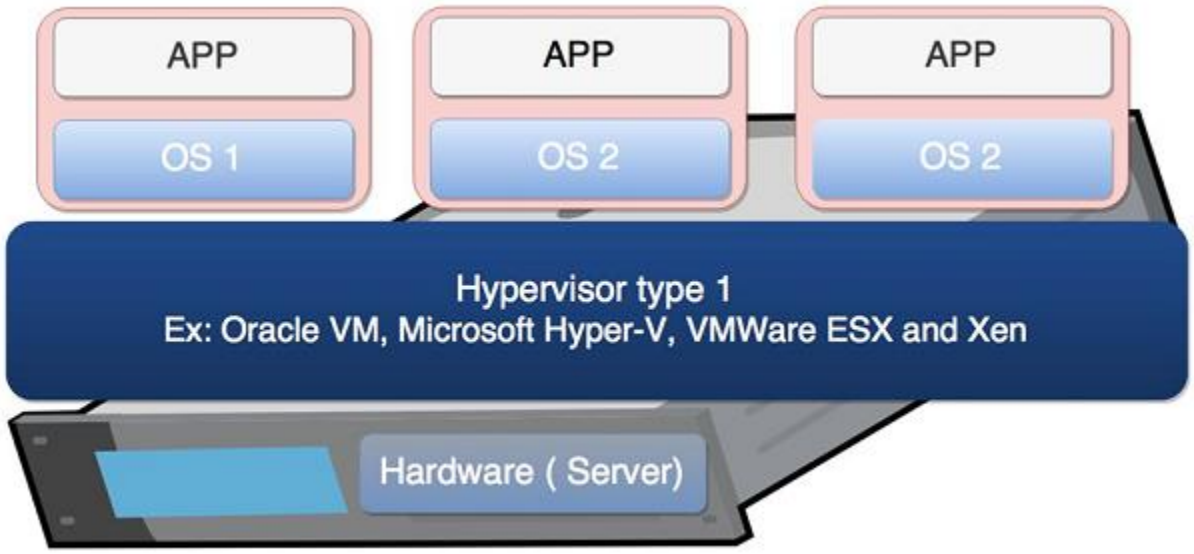
Let us now discuss both of these in detail.

Native or Bare Metal Hypervisor

Native hypervisors are software systems that run directly on the host's hardware to control the hardware and to monitor the **Guest Operating Systems**. The guest

operating system runs on a separate level above the hypervisor. All of them have a Virtual Machine Manager.

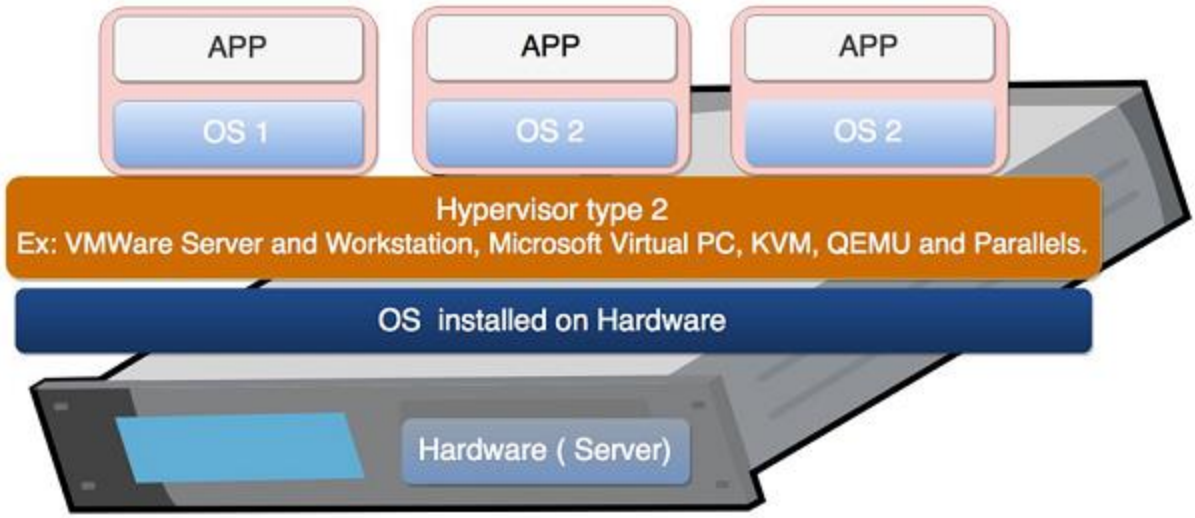
Examples of this virtual machine architecture are **Oracle VM, Microsoft Hyper-V, VMWare ESX** and **Xen**.



Hosted Hypervisor

Hosted hypervisors are designed to run within a traditional operating system. In other words, a hosted hypervisor adds a distinct software layer on top of the host operating system. While, the guest operating system becomes a third software level above the hardware.

A well-known example of a hosted hypervisor is **Oracle VM VirtualBox**. Others include **VMWare Server and Workstation, Microsoft Virtual PC, KVM, QEMU**and **Parallels**.



Understanding Local Virtualization and Cloud

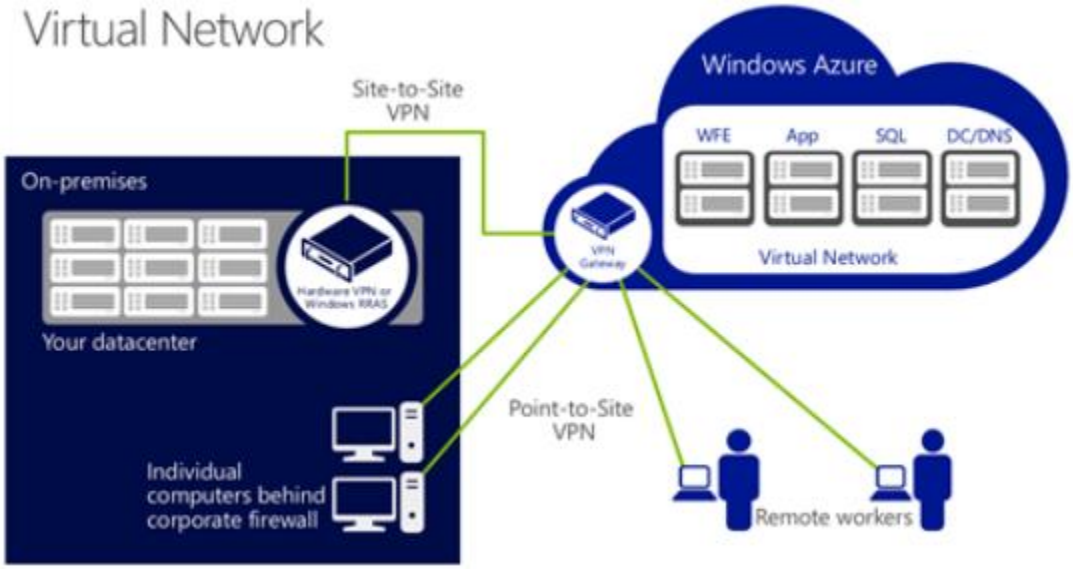
Virtualization is one of the fundamental technologies that makes cloud-computing work. However, virtualization is not cloud computing. Cloud computing is a service that different providers offer to you based on some costs.

In enterprise networks, virtualization and cloud computing are often used together to build a public or private cloud infrastructure. In small businesses, each technology will be deployed separately to gain measurable benefits. In different ways, virtualization and cloud computing can help you keep your equipment spending to a minimum and get the best possible use from the equipment you already have.

As mentioned before, virtualization software allows one physical server to run several individual computing environments. In practice, it is like getting multiple servers for each physical server you buy. This technology is fundamental to cloud computing. Cloud providers have large data centers full of servers to power their cloud offerings, but they are not able to devote a single server to each customer. Thus, they virtually partition the data on the server, enabling each client to work with a separate “virtual” instance (which can be a private network, servers farm, etc.) of the same software.

Small businesses are most likely to adopt cloud computing by subscribing to a cloud-based service. The largest providers of cloud computing are **Microsoft with Azure** and **Amazon**.

The following illustration is provided by Microsoft where you can understand how utilizing extra infrastructure for your business without the need to spend extra money helps. You can have the on-premises base infrastructure, while on cloud you can have all your services, which are based on Virtualized technology.



ADVACED WEB TECHNOLOGIES:

A mashup is created when several data sources and services are "mashed up" (combined) to create something new, or add value in some way. This tutorial shows how to create a Google Maps mashup—combining the mapping data supplied by Google Maps with a location data service that you can build yourself.
The tutorial is structured in three parts:

- The Web 2.0 Toolbox
- Retrieving Locations from a REST Data Service

- Plotting Addresses with Google Maps

The first section covers the technology that I will use to build the mashup in the second and third sections. For a look ahead at the demo that I will build, navigate to the hosted demo on the Web. Let's get started!

The Web 2.0 Toolbox

Web developers know that technology is always evolving, and skills can become obsolete in just a few years. I wouldn't have it any other way: innovation is what makes Web development interesting. New ideas, new tools, and new technologies allow us to build better systems in a shorter amount of time. This section covers a number of technologies that you may or may not have used. If these are new to you, consider this section a quick introduction, and I would encourage you to follow up with deeper study. The descriptions below are by no means complete, and in some cases overly simplified so as to convey the important concepts.

While I will discuss a handful of technologies, this tutorial has to start somewhere. It is assumed that you are comfortable with the following concepts and Web technologies:

- HTML
- XML
- The role of browsers and Web servers
- The HTTP request/response model
- A modern programming language, like Java, JavaScript, PHP, C#

Client-side Programming

A hallmark of a Web 2.0 application is that it is highly interactive—responsive like a traditional desktop application. The staid feel of a traditional Web application, with lengthy pauses for full-page refreshes, cannot live up to this expectation. Therefore, Web 2.0 applications make use of client-side programming technologies to help make the application feel more responsive. The two most popular client-side technologies are JavaScript and Adobe Flex. Both have compelling features and both can be used with great success. However, to narrow the scope of this tutorial, I will look only at the JavaScript approach when building the mashup.

JavaScript is no toy. It is a powerful client-side programming language that has certainly come of age. Cross-browser support has improved dramatically as standards have evolved, making it a viable choice. For those new to the language, you will find its basic code constructs similar to other mainstream languages like Java.
A powerful *event mechanism* is included to enable JavaScript to respond to user interactions in the browser. I will rely on the eventing capabilities to build the mashup later in the tutorial. Students of HTML have likely already seen the JavaScript eventing mechanism in the form of events such as the onclick attribute, as in:
Copy

Copied to Clipboard

Error: Could not Copy

<onclick="javascript:myEventHandler(); return true"

href="myURL.html">My Link

Another important feature of JavaScript when executed in the browser is its ability to manipulate the HTML Document Object Model (DOM). This feature allows JavaScript code to programmatically alter the contents of an HTML page after it has been loaded. **DOM manipulation** is an important feature to use when increasing the interactivity of a Web 2.0 application. In Web applications, it is common to reset the text contained by an element in the HTML, as in:

Copy

Copied to Clipboard

Error: Could not Copy

// find the tag with id 'greet_div'

var div = document.getElementById('greet_div');

div.innerHTML = 'Hello ' + name;

The final JavaScript feature that must be mentioned is its ability to issue out-of-band HTTP requests to back-end servers. As a result, JavaScript can issue requests that do not cause the page to reload or change the address bar of the browser. This feature is commonly called *Ajax*, but the source of this feature is the XMLHttpRequest which is the JavaScript class that can invoke the HTTP request. The HTTP requests generally are asynchronous, which requires the programmer to define a callback function to be called when the response is received.

Copy

Copied to Clipboard

Error: Could not Copy

var request = new XMLHttpRequest();

```
function invokeAjax() {
        request.open("GET", 'ajaxTarget.html', true);
        request.onreadystatechange = ajaxCallback;
        request.send(null);
}


function ajaxCallback() {
  // check if response is complete, then do stuff
}
```

I cannot consider the XMLHttpRequest feature covered until I discuss one major limitation that will affect its ability to be used in mashup applications.
To protect users from malicious code writers, a safety feature is present in all browser implementations. The Same Origin Policy prevents an XMLHttpRequest from targeting a server in any network domain other than the network domain that provided the page. For example, if the user browses to http://www.bea.com/ajaxPage.html, the JavaScript code on that page could not issue an XMLHttpRequest to http://www.evil.com/stealCookies.html. While this safety feature protects users, it limits the role of client-side JavaScript programming in mashup applications that need to consume services from multiple domains. However, two resource types are immune to this policy: a page can load cross-domain images and scripts. By being clever with appending parameters to these resource requests, some implementations have worked around the limitation.
For more details on Ajax and XMLHttpRequest, see An Introduction To Ajax (Dev2Dev, 2005).

Lightweight Services

The ability to invoke a remote service is the cornerstone of distributed architectures in the enterprise like Service Oriented Architecture (SOA). Web service technologies such as SOAP have been widely adopted to create reusable services within the enterprise. These implementations work well, but in some cases SOAP is overkill. Especially when the client is a browser, a lighter-weight solution is wanted.

An approach to building lightweight services called REST has become popular, especially with Web 2.0 applications. REST offers a clean model for building HTTP addressable services that

are easily invoked from a browser. While a full academic definition of REST is not appropriate for this tutorial, I can summarize with a few bullet points:

- The REST service is expressed as a URL and is accessed with basic HTTP requests, such as http://bea.com/content/getArticles?author=joe.
- The HTTP verb is important: a GET is a read operation, POST is a creation, and PUT make updates to the service.
- The return payload is usually XML or JSON.

Trying to be any more specific to the meaning of REST will likely breed controversy, so the above characterization will suffice.

The last bullet point requires further explanation. Two options are listed as popular formats for the return payload: XML and JSON. XML would seem to be the format of choice, as it is widely used throughout the world. However, while REST services can certainly return XML, the client JavaScript code will need to walk the DOM of the returned XML to extract the information it needs. This is okay, but for cases where the client will likely be JavaScript in a browser, another option exists. **JavaScript Object Notation** (JSON) is a JavaScript object serialization format that eases the job of the client. The client can make one call to deserialize the returned JSON text into a native JavaScript object, which then can be manipulated using standard JavaScript syntax. This is often an easier approach, and therefore JSON is popular for REST services that are consumed by Web 2.0 applications.
Here is an example of a JSON object in serialized form:

Copy

Copied to Clipboard

Error: Could not Copy

```
{"location":
      {"id":  "WashingtonDC",
       "city": "Washington DC",
       "venue": "Hilton Hotel, Tysons Corner",
       "address": "7920 Jones Branch Drive"
      }
}
```

Architectural aspects of mashups

The architecture of a mashup is divided into three layers:

- Presentation / user interaction: this is the user interface of mashups. The technologies used are HTML/XHTML, CSS, JavaScript, Asynchronous JavaScript and Xml (Ajax).
- Web Services: the product's functionality can be accessed using API services. The technologies used are XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.
- Data: handling the data like sending, storing and receiving. The technologies used are XML, JSON, KML.

Architecturally, there are two styles of mashups: Web-based and server-based. Whereas Web-based mashups typically use the user's web browser to combine and reformat the data, server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form.[10]

Mashups appear to be a variation of a façade pattern.[11] That is: a software engineering design pattern that provides a simplified interface to a larger body of code (in this case the code to aggregate the different feeds with different APIs).

Mashups can be used with software provided as a service (SaaS).

After several years of standards development, mainstream businesses are starting to adopt service-oriented architectures (SOA) to integrate disparate data by making them available as discrete Web services. Web services provide open, standardized protocols to provide a unified means of accessing information from a diverse set of platforms (operating systems, programming languages, applications). These Web services can be reused to provide completely new services and applications within and across organizations, providing business flexibility.