Cloud providers Overview

Cloud Computing provides us means of accessing the applications as utilities over the Internet. It allows us to create, configure, and customize the applications online.

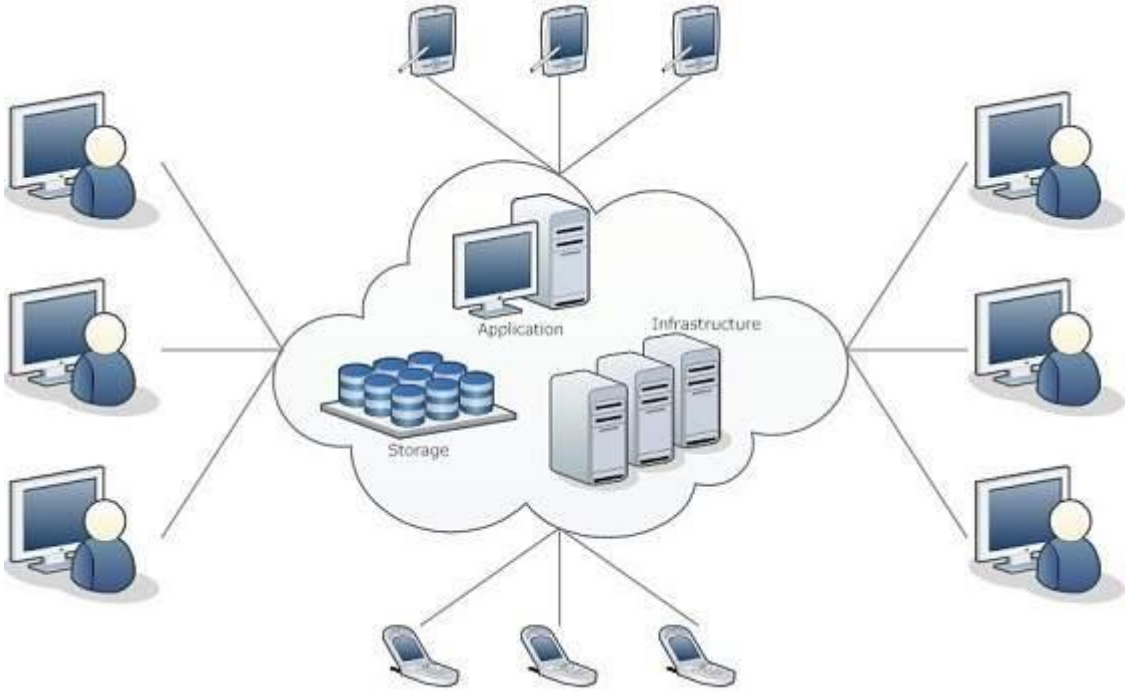
What is Cloud?

The term **Cloud** refers to a **Network** or **Internet.** In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN or VPN.

Applications such as e-mail, web conferencing, customer relationship management (CRM) execute on cloud.

What is Cloud Computing?

Cloud Computing refers to **manipulating, configuring,** and **accessing** the hardware and software resources remotely. It offers online data storage, infrastructure, and application.



Cloud computing offers **platform independency,** as the software is not required to be installed locally on the PC. Hence, the Cloud Computing is making our business applications **mobile** and **collaborative.**

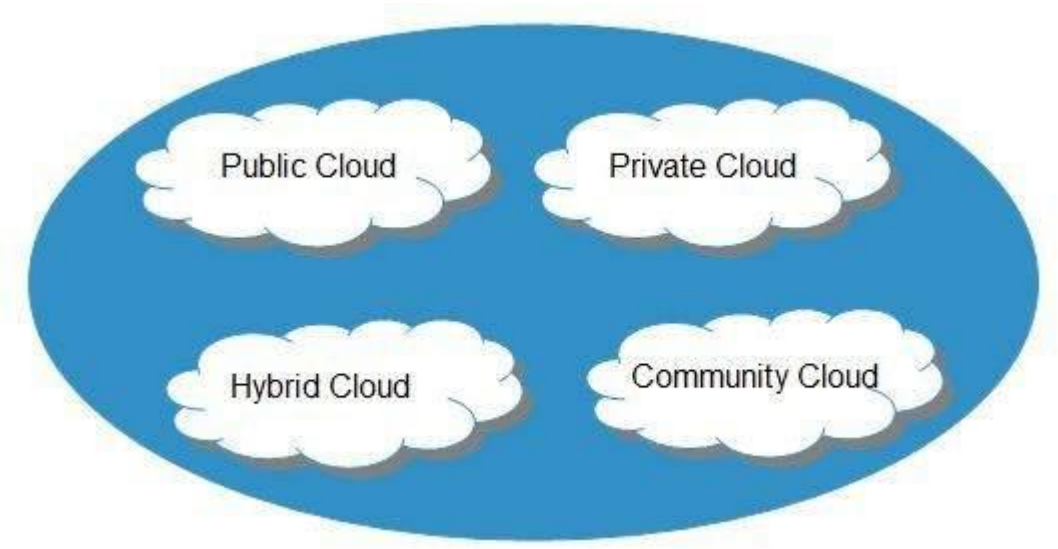
Basic Concepts

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

- Deployment Models
- Service Models

Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid, and Community.



Public Cloud

The **public cloud** allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness.

Private Cloud

The **private cloud** allows systems and services to be accessible within an organization. It is more secured because of its private nature.

Community Cloud

The **community cloud** allows systems and services to be accessible by a group of organizations.

Hybrid Cloud

The **hybrid cloud** is a mixture of public and private cloud, in which the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

Service Models

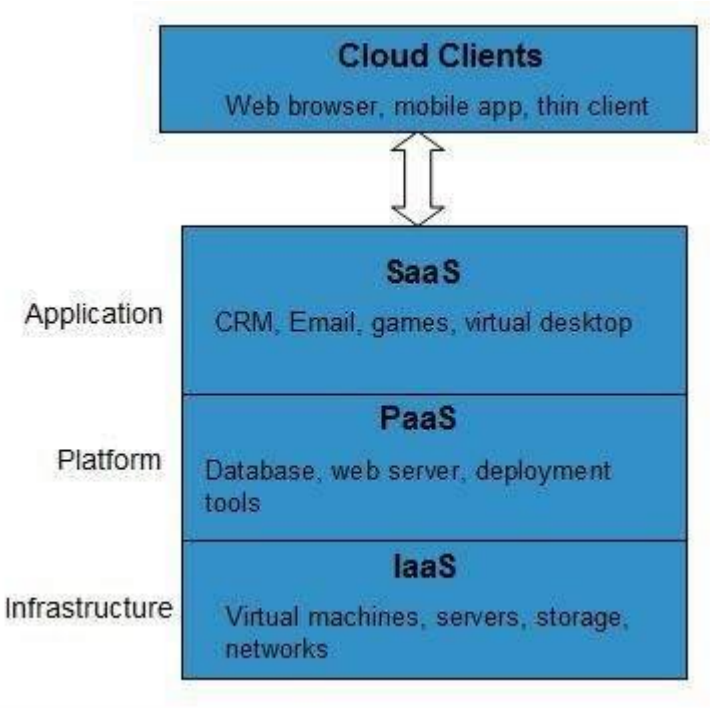
Cloud computing is based on service models. These are categorized into three basic service models which are -

- Infrastructure-as–a-Service (IaaS)

- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Anything-as-a-Service (XaaS) is yet another service model, which includes Network-as-a-Service, Business-as-a-Service, Identity-as-a-Service, Database-as-a-Service or Strategy-as-a-Service.

The **Infrastructure-as-a-Service (IaaS)** is the most basic level of service. Each of the service models inherit the security and management mechanism from the underlying model, as shown in the following diagram:



Infrastructure-as-a-Service (IaaS)

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

Platform-as-a-Service (PaaS)

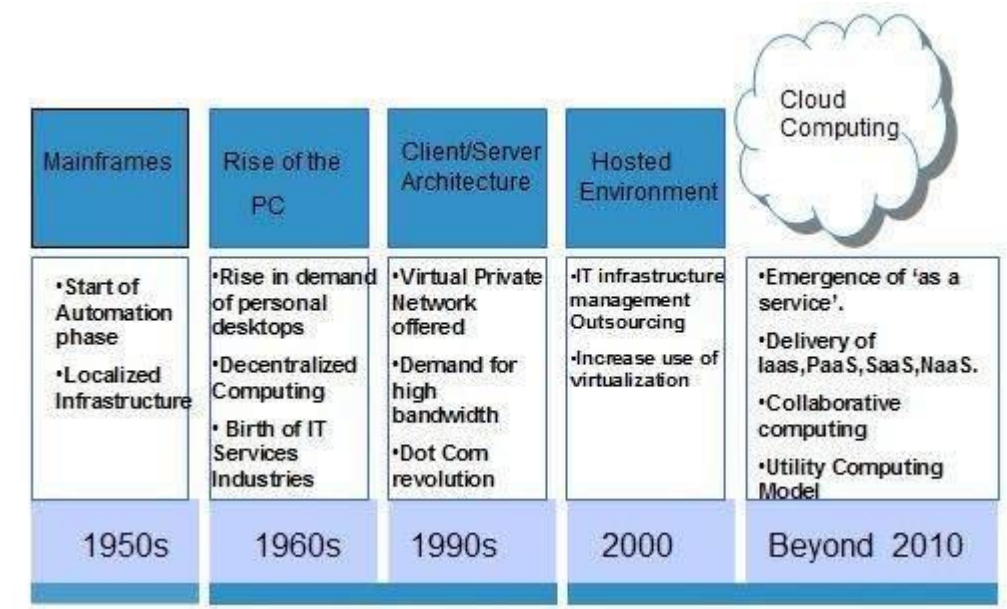
PaaS provides the runtime environment for applications, development and deployment tools, etc.

Software-as-a-Service (SaaS)

SaaS model allows to use software applications as a service to end-users.

History of Cloud Computing

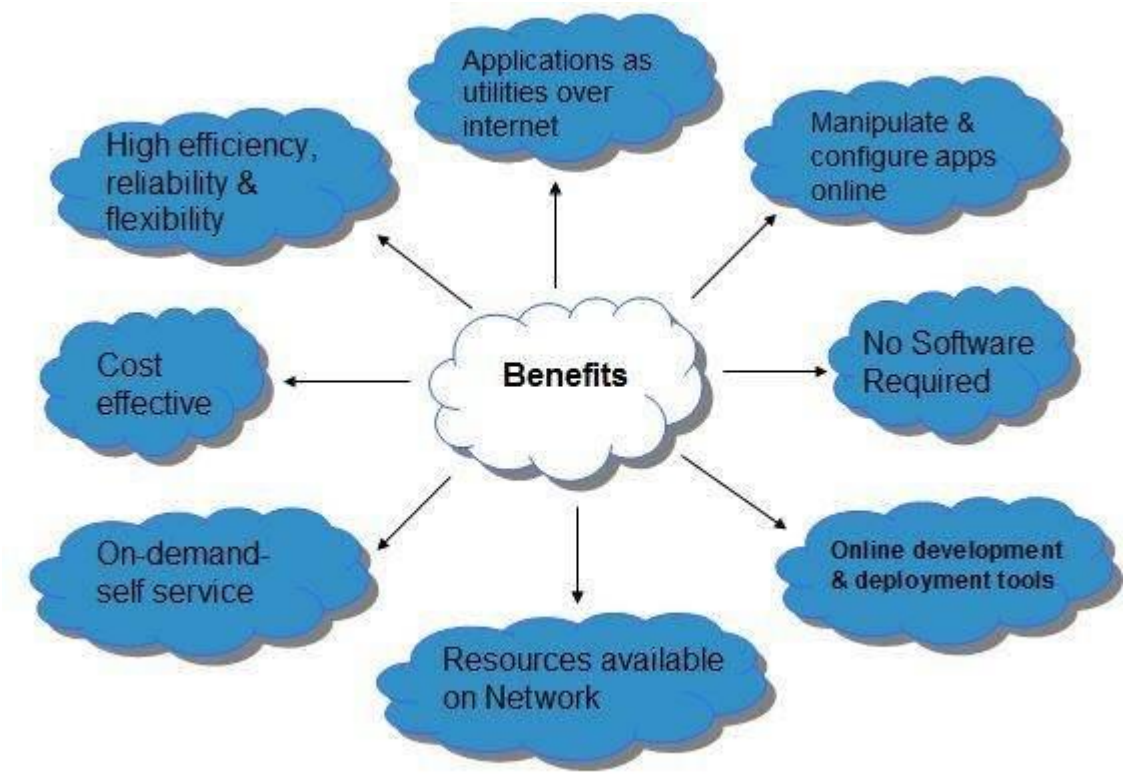
The concept of **Cloud Computing** came into existence in the year 1950 with implementation of mainframe computers, accessible via **thin/static clients.** Since then, cloud computing has been evolved from static clients to dynamic ones and from software to services. The following diagram explains the evolution of cloud computing:



Benefits

Cloud Computing has numerous advantages. Some of them are listed below -

- One can access applications as utilities, over the Internet.
- One can manipulate and configure the applications online at any time.
- It does not require to install a software to access or manipulate cloud application.
- Cloud Computing offers online development and deployment tools, programming runtime environment through **PaaS model.**
- Cloud resources are available over the network in a manner that provide platform independent access to any type of clients.
- Cloud Computing offers **on-demand self-service.** The resources can be used without interaction with cloud service provider.
- Cloud Computing is highly cost effective because it operates at high efficiency with optimum utilization. It just requires an Internet connection
- Cloud Computing offers load balancing that makes it more reliable.



Risks related to Cloud Computing

Although cloud Computing is a promising innovation with various benefits in the world of computing, it comes with risks. Some of them are discussed below:

Security and Privacy

It is the biggest concern about cloud computing. Since data management and infrastructure management in cloud is provided by third-party, it is always a risk to handover the sensitive information to cloud service providers.

Although the cloud computing vendors ensure highly secured password protected accounts, any sign of security breach may result in loss of customers and businesses.

Lock In

It is very difficult for the customers to switch from one **Cloud Service Provider (CSP)** to another. It results in dependency on a particular CSP for service.

Isolation Failure

This risk involves the failure of isolation mechanism that separates storage, memory, and routing between the different tenants.

Management Interface Compromise

In case of public cloud provider, the customer management interfaces are accessible through the Internet.

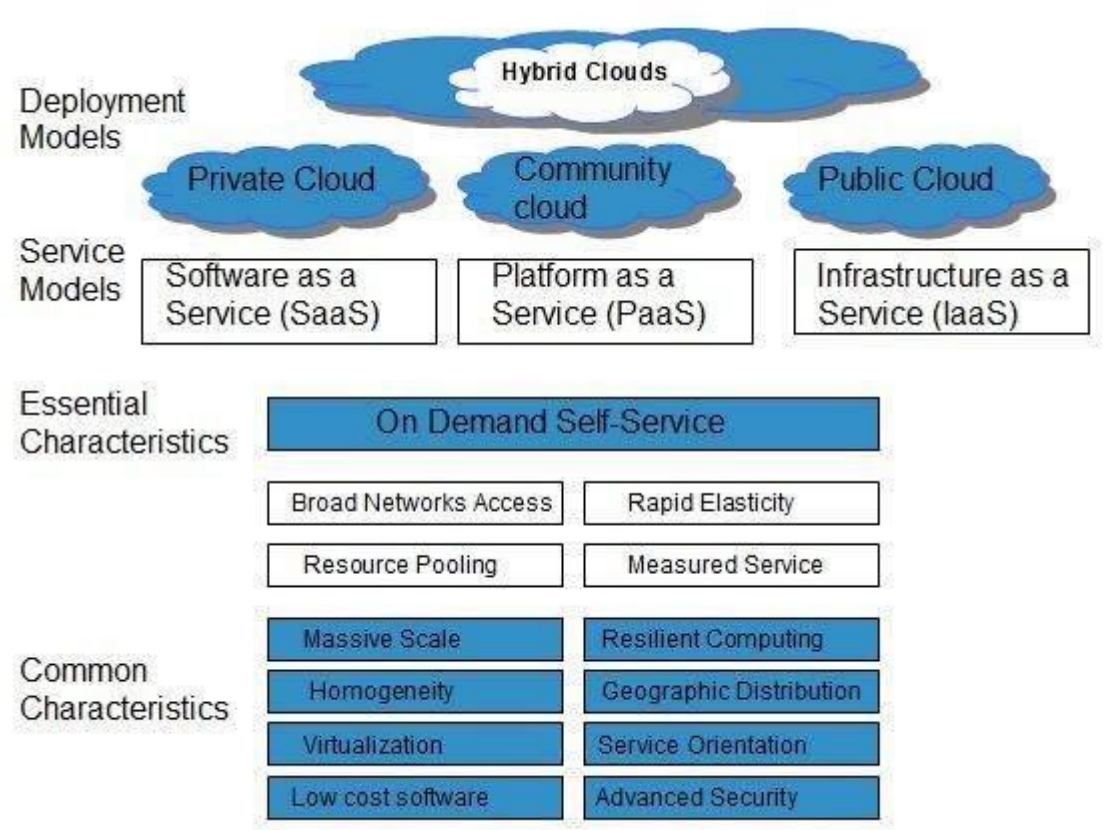
Insecure or Incomplete Data Deletion

It is possible that the data requested for deletion may not get deleted. It happens because either of the following reasons

- Extra copies of data are stored but are not available at the time of deletion
- Disk that stores data of multiple tenants is destroyed.

Characteristics of Cloud Computing

There are four key characteristics of cloud computing. They are shown in the following diagram:



On Demand Self Service

Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them.

Broad Network Access

Since cloud computing is completely web based, it can be accessed from anywhere and at any time.

Resource Pooling

Cloud computing allows multiple tenants to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.

Rapid Elasticity

It is very easy to scale the resources vertically or horizontally at any time. Scaling of resources means the ability of resources to deal with increasing or decreasing demand.

The resources being used by customers at any given point of time are automatically monitored.

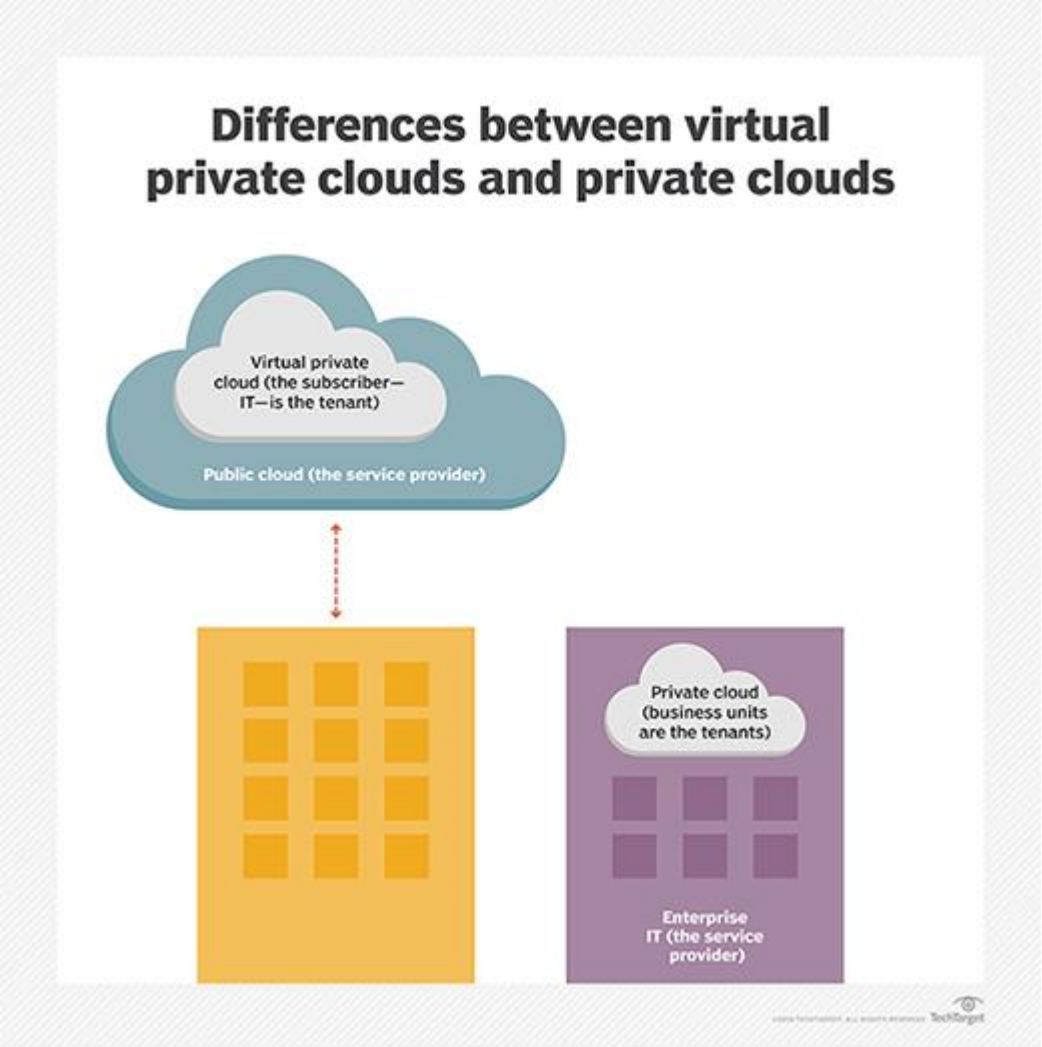
Measured Service

In this service cloud provider controls and monitors all the aspects of cloud service. Resource optimization, billing, and capacity planning etc. depend on it.

Virtual private cloud (VPC)

A virtual private cloud (VPC) is the logical division of a service provider's public cloud multi-tenant architecture to support private cloud computing. This model enables an enterprise to achieve the benefits of private cloud -- such as more granular control over virtual networks and an isolated environment for sensitive workloads -- while still taking advantage of public cloud resources.

The terms *private cloud* and *virtual private cloud* are sometimes used incorrectly as synonyms. There is a distinct difference -- in a traditional, on-premises private cloud model, an enterprise's internal IT department acts as a service provider and the individual business units act as tenants. With a VPC, a public cloud provider acts as the service provider and the cloud's subscribers are the tenants.



How a virtual private cloud works

In a virtual private cloud model, the public infrastructure-as-a-service (IaaS) provider is responsible for ensuring that each private cloud customer's data remains isolated from every other customer's data both in transit and inside the cloud provider's network. This can be accomplished through the use of security policies requiring some -- or all -- of the following elements: encryption, tunneling, private IP addressing or allocating a unique virtual local area network (VLAN) to each customer.

A virtual private cloud user can define and directly manage network components, including IP addresses, subnets, network gateways and access control policies.

Benefits and challenges of virtual private clouds

As mentioned above, one of the biggest benefits of VPCs is that they enable an enterprise to tap into some of the benefits of private clouds, such as more granular network control, while still using off-premises, public cloud resources in a highly scalable, pay-as-you-go model.

Another benefit of VPCs is enabling a hybrid cloud deployment. An enterprise can use a VPC as an extension of its own data center without dealing with the complexities of building an on-premises private cloud.

Despite the benefits of VPCs, they can also introduce some challenges. For example, an enterprise might face some complexity when configuring, managing and monitoring its virtual private network (VPN).

In addition, while VPCs offer an isolated environment within a public cloud in which workloads can run, they are still hosted outside an enterprise's own data center. This means that businesses in highly regulated industries with strict compliance requirements might face limitations on which kinds of applications and data they can place in a VPC.

Before it commits to a VPC, an enterprise should also verify that all of the resources and services it wants to use from its chosen public cloud provider are available via that provider's VPC.

Virtual private cloud providers

Most leading public IaaS providers, including Amazon Web Services (AWS), Microsoft Azure and Google, offer VPC and virtual network services.

Scaling in Cloud Computing

Cloud scalability in cloud computing refers to increasing or decreasing IT resources as needed to meet changing demand. Scalability is one of the hallmarks of the cloud and the primary driver of its explosive popularity with businesses.

Data storage capacity, processing power, and networking can all be increased by using existing cloud computing infrastructure. Scaling can be done quickly and easily, usually without any disruption or downtime.

Third-party cloud providers already have the entire infrastructure in place; In the past, when scaling up with on-premises physical infrastructure, the process could take weeks or months and require exorbitant expenses.

This is one of the most popular and beneficial features of cloud computing, as businesses can grow up or down to meet the demands depending on the season, projects, development, etc.

By implementing cloud scalability, you enable your resources to grow as your traffic or organization grows and vice versa. There are a few main ways to scale to the cloud:

If your business needs more data storage capacity or processing power, you'll want a system that scales easily and quickly.

Cloud computing solutions can do just that, which is why the market has grown so much. Using existing cloud infrastructure, third-party cloud vendors can scale with minimal disruption.

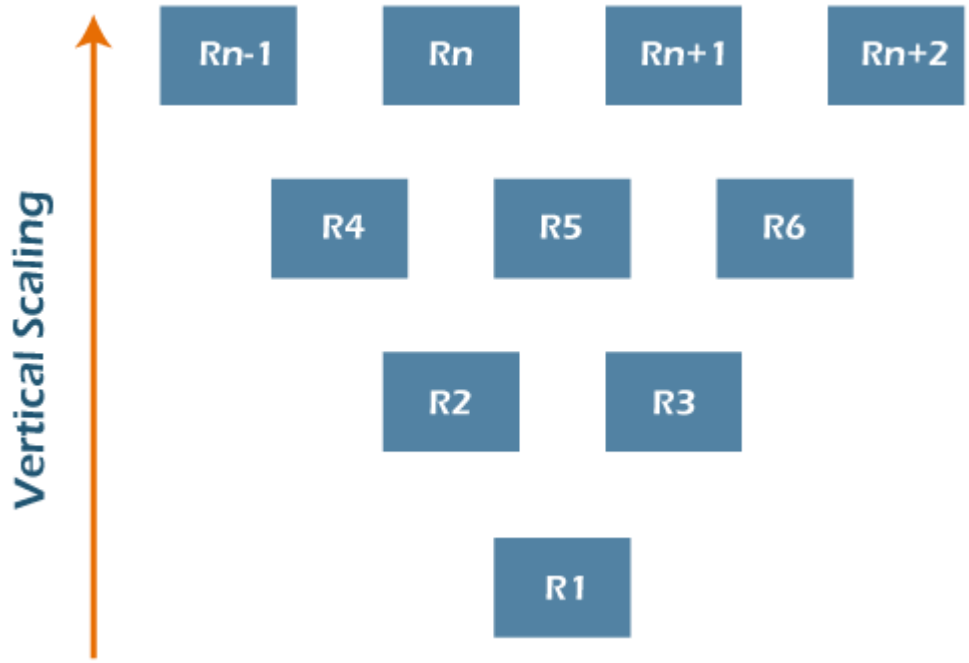
Types of scaling

- Vertical Scalability (Scaled-up)
- horizontal scalability
- diagonal scalability

Vertical Scaling

To understand vertical scaling, imagine a 20-story hotel. There are innumerable rooms inside this hotel from where the guests keep coming and going. Often there are spaces available, as not all rooms are filled at once. People can move easily as there is space for them. As long as the capacity of this hotel is not exceeded, no problem. This is vertical scaling.

With computing, you can add or subtract resources, including memory or storage, within the server, as long as the resources do not exceed the capacity of the machine. Although it has its limitations, it is a way to improve your server and avoid latency and extra management. Like in the hotel example, resources can come and go easily and quickly, as long as there is room for them.



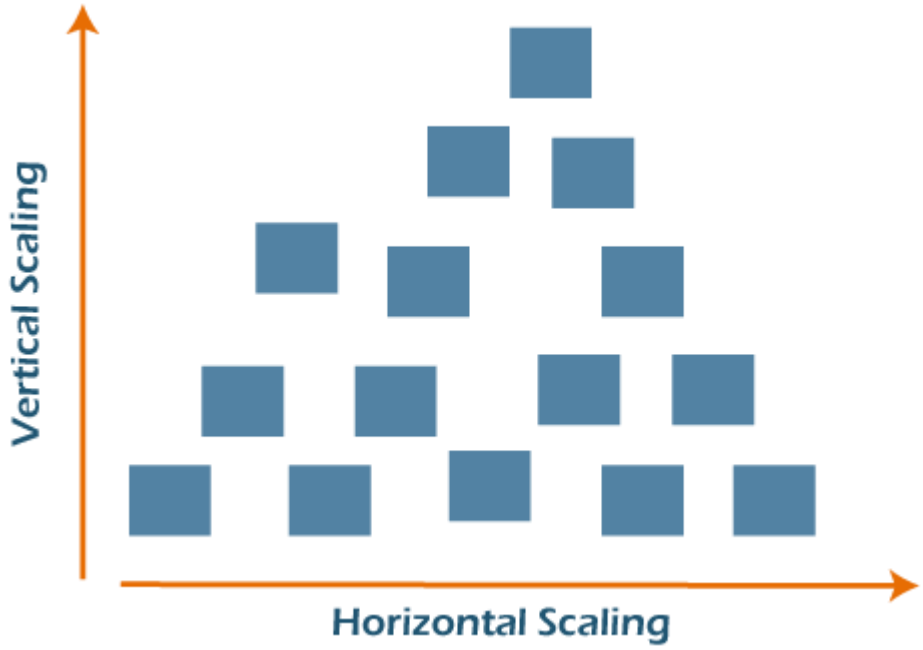
Horizontal Scaling

Horizontal scaling is a bit different. This time, imagine a two-lane highway. Cars travel smoothly in each direction without major traffic problems. But then the area around the highway develops - new buildings are built, and traffic increases. Very soon, this two-lane highway is filled with cars, and accidents become common. Two lanes are no longer enough. To avoid these issues, more lanes are added, and an overpass is constructed. Although it takes a long time, it solves the problem.

Horizontal scaling refers to adding more servers to your network, rather than simply adding resources like with vertical scaling. This method tends to take more time and is more complex, but it allows you to connect servers together, handle traffic efficiently and execute concurrent workloads.

Diagonal Scaling

It is a mixture of both Horizontal and Vertical scalability where the resources are added both vertically and horizontally. Well, you get diagonal scaling, which allows you to experience the most efficient infrastructure scaling. When you combine vertical and horizontal, you simply grow within your existing server until you hit the capacity. Then, you can clone that server as necessary and continue the process, allowing you to deal with a lot of requests and traffic concurrently.



Scale in the Cloud

When you move scaling into the cloud, you experience an enormous amount of flexibility that saves both money and time for a business. When your demand booms, it's easy to scale up to accommodate the new load. As things level out again, you can scale down accordingly.

This is so significant because cloud computing uses a pay-as-you-go model.

Traditionally, professionals guess their maximum capacity needs and purchase everything up front. If they overestimate, they pay for unused resources.

If they underestimate, they don't have the services and resources necessary to operate effectively. With cloud scaling, though, businesses get the capacity they need when they need it, and they simply pay based on usage. This on-demand nature is what makes the

cloud so appealing. You can start small and adjust as you go. It's quick, it's easy, and you're in control.

Difference between Cloud Elasticity and Scalability:

Cloud Elasticity	Cloud Scalability
Elasticity is used just to meet the sudden up and down in the workload for a small period of time.	Scalability is used to meet the static increase in the workload.
Elasticity is used to meet dynamic changes, where the resources need can increase or decrease.	Scalability is always used to address the increase in workload in an organization.
Elasticity is commonly used by small companies whose workload and demand increases only for a specific period of time.	Scalability is used by giant companies whose customer circle persistently grows in order to do the operations efficiently.
It is a short term planning and adopted just to deal with an unexpected increase in demand or seasonal demands.	Scalability is a long term planning and adopted just to deal with an expected increase in demand.

Why is cloud scalable?

Scalable cloud architecture is made possible through virtualization. Unlike physical machines whose resources and performance are relatively set, virtual machines virtual machines (VMs) are highly flexible and can be easily scaled up or down. They can be moved to a different server or hosted on multiple servers at once; workloads and applications can be shifted to larger VMs as needed.

Third-party cloud providers also have all the vast hardware and software resources already in place to allow for rapid scaling that an individual business could not achieve cost-effectively on its own.

Benefits of cloud scalability

Key cloud scalability benefits driving cloud adoption for businesses large and small:

- **Convenience**: Often, with just a few clicks, IT administrators can easily add more VMs that are available-and customized to an organization's exact needs-without delay. Teams can focus on other tasks instead of setting up physical hardware for hours and days. This saves the valuable time of the IT staff.
- **Flexibility and speed**: As business needs change and grow, including unexpected demand spikes, cloud scalability allows IT to respond quickly. Companies are no longer tied to obsolete equipment-they can update systems and easily increase power and storage. Today, even small businesses have access to high-powered resources that used to be cost-prohibitive.
- **Cost Savings**: Thanks to cloud scalability, businesses can avoid the upfront cost of purchasing expensive equipment that can become obsolete in a few years. Through cloud providers, they only pay for what they use and reduce waste.
- **Disaster recovery**: With scalable cloud computing, you can reduce disaster recovery costs by eliminating the need to build and maintain secondary data centers.

When to Use Cloud Scalability?

Successful businesses use scalable business models to grow rapidly and meet changing demands. It's no different with their IT. Cloud scalability benefits help businesses stay agile and competitive.

Scalability is one of the driving reasons for migrating to the cloud. Whether traffic or workload demands increase suddenly or increase gradually over time, a scalable cloud solution enables organizations to respond appropriately and cost-effectively to increased storage and performance.

How do you determine optimal cloud scalability?

Changing business needs or increasing demand often necessitate your scalable cloud solution changes. But how much storage, memory, and processing power do you need? Will you scale in or out?

To determine the correct size solution, continuous performance testing is essential. IT administrators must continuously measure response times, number of requests, CPU load, and memory usage. Scalability testing also measures the performance of an application and its ability to scale up or down based on user requests.

Automation can also help optimize cloud scalability. You can set a threshold for usage that triggers automatic scaling so as not to affect performance. You may also consider a third-party configuration management service or tool to help you manage your scaling needs, goals, and implementation.

Virtualization in Cloud Computing

Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".

In other words, Virtualization is a technique, which allows to share a single physical instance of a resource or an application among multiple customers and organizations. It does by assigning a logical name to a physical storage and providing a pointer to that physical resource when demanded.

What is the concept behind the Virtualization?

Creation of a virtual machine over existing operating system and hardware is known as Hardware Virtualization. A Virtual machine provides an environment that is logically separated from the underlying hardware.

The machine on which the virtual machine is going to create is known as **Host Machine** and that virtual machine is referred as a **Guest Machine**

Types of Virtualization:

1. Hardware Virtualization.
2. Operating system Virtualization.
3. Server Virtualization.
4. Storage Virtualization.

1) Hardware Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is directly installed on the hardware system* is known as hardware virtualization.

The main job of hypervisor is to control and monitoring the processor, memory and other hardware resources.

After virtualization of hardware system we can install different operating system on it and run different applications on those OS.

Usage:

Hardware virtualization is mainly done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

2) Operating System Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is installed on the Host operating system* instead of directly on the hardware system is known as operating system virtualization.

Usage:

Operating System Virtualization is mainly used for testing the applications on different platforms of OS.

3) Server Virtualization:

When the virtual machine software or virtual machine manager *(VMM) is directly installed on the Server system* is known as server virtualization.

Usage:

Server virtualization is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage Virtualization:

Storage virtualization is the *process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device*.

Storage virtualization is also implemented by using software applications.

Usage:

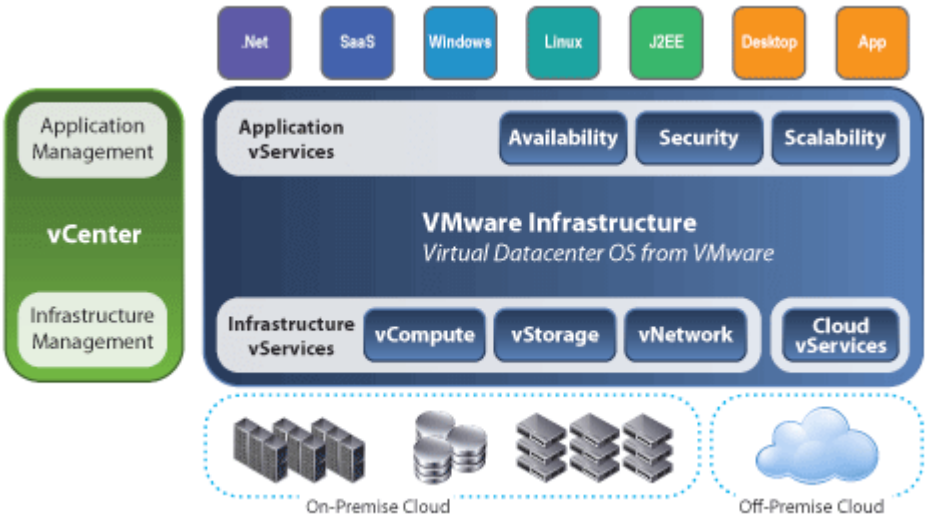
Storage virtualization is mainly done for back-up and recovery purposes.

How does virtualization work in cloud computing?

Virtualization plays a very important role in the cloud computing technology, normally in the cloud computing, users share the data present in the clouds like application etc, but actually with the help of virtualization users shares the Infrastructure.

The **main usage of Virtualization Technology** is to provide the applications with the standard versions to their cloud users, suppose if the next version of that application is released, then cloud provider has to provide the latest version to their cloud users and practically it is possible because it is more expensive.

To overcome this problem we use basically virtualization technology, By using virtualization, all severs and the software application which are required by other cloud providers are maintained by the third party people, and the cloud providers has to pay the money on monthly or annual basis.



What Is the Ethernet?

The Ethernet connects equipment such as switches, routers, and computers using a simple user interface. Communication between the linked devices is enabled using a local area network (LAN) using a single router and some Ethernet cables. The Ethernet is different from the Internet, which connects computers utilizing a telephone line, software protocol, and hardware. WiFi sends data using wireless signals.

What Is the Ethernet: How the Ethernet Was Born

The Ethernet was created in 1973 at Xerox's Palo Alto Research Center (PARC)

by Robert Metcalfe and others. Xerox patented the technology in 1975. Metcalfe had two challenges: to build a fast enough network to drive the company's new laser printer and to connect hundreds of computers in the same building. An open Ethernet standard was finalized in 1980, and by 1985, it had become an Institute of Electrical and Electronics Engineers (IEEE) standard. IEEE defines Ethernet as protocol 802.3.

Modern PCs began including Ethernet cards on the motherboard, which became very inexpensive. Ethernet networks in the workplace started with some small companies using telephone-based, four-wire lines. It wasn't until the early 1990s that an Ethernet connection was established using twisted pair and fiber optic cables. This development facilitated the introduction of the 100 MB/s standard in 1995.

What Is the Ethernet: How the Ethernet Works

The Ethernet facilitates the operation of physical and data link layers and resides in the Open Systems Interconnection (OSI) lower layers. OSI is a model describing how information from a software application on one computer moves to the software application on another computer.

The OSI model has seven layers:

1. Physical Layer – establishes, maintains, and deactivates the physical connection. Its primary function is transmitting the individual bits from one node to another.
2. Data-Link Layer – responsible for the error-free transfer of data frames. It is responsible for uniquely identifying each device that resides on a local network.
3. Network Layer – manages device addressing and tracks the location of devices on the network. It selects the best path to move data from a source to the destination based on the network conditions, service priority, and other factors.
4. Transport Layer – ensures that messages are transmitted in the correct order and that there is no data duplication. Its main task is to transfer the data completely.

5. Session Layer – establishes, maintains, and synchronizes the interaction between communicating devices.
6. Presentation Layer – serves as the network's data translator and works with the syntax and semantics of the information exchanged between two systems.
7. Application Layer – helps users and application processes access network service.

Ethernet Speeds

The early Ethernet speeds puttered a mere 10 megabits per second (Mbps). Fast Ethernet increased data transfers up to 100 Mbps, and Gigabit Ethernet boasts speeds of up to 1,000 Mbps. In 2002, 10 Gigabit Ethernet was introduced with speeds of up to 10,000 Mbps. According to TechTarget, 10 Gigabit Ethernet provides a cost-effective means of addressing the speed requirements of newer computing applications — streaming, data centers, video, virtualization, data backups, and high-end gaming.

What Are Ethernet Cables and Setup

Not all Ethernet cables are alike. The most common Ethernet cable is Category 5 (or CAT5), which supports traditional and Fast Ethernet. Category 5e and Category 6 (CAT5e and CAT6) cables are used for Gigabit and 10 Gigabit Ethernet.

Ethernet cables run from the modem or modem-router combo (or gateway) to the Ethernet ports on devices such as desktop computers, laptop computers, and televisions.

Different Types of Ethernet Networks

An Ethernet network usually is active in a 10-kilometer periphery, according to Versitron. Using fiber optic cable increases the distance covered by a network. Ethernet networks include:

- Fast Ethernet: This high-speed network can send or receive data at about 100 Mbps. Fast Ethernet has three categories: 100BASE-TX, 100BASE-FX, and 100BASE-T4.

- Gigabit Ethernet: This network is one of the most widely used, and it transfers data at 1000 Mbps or 1Gbps. Gigabit Ethernet has fostered a faster transfer of data and a faster communication network.
- 10-Gigabit Ethernet: This advanced, high-speed network transmits data at a rate of 10 Gigabit/second. The network can be extended to 10,000 meters using fiber-optic cable.
- Switch Ethernet: This network has added switches or hubs, which can improve network throughput. It allows each workstation to have a dedicated 10 Mbps connection instead of sharing. Switch Ethernet supports 1000Mbps to 10 Gbps and 10Mbps to 100Mbps for Fast Ethernet.

Cloud models map to application and connectivity requirements. The most common cloud offerings are SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). Public cloud services (e.g., AWS, Google, etc.) support shared applications, accommodate distributed users and are generally accessible via the Internet. Private clouds are designed for use by a single entity and accommodate business-critical applications with specific security or performance requirements. Private clouds reside at enterprise data centers, off-site data centers or collocation facilities, or are managed by a cloud service provider (e.g., AT&T, CenturyLink Savvis, Verizon Terremark, etc.). Connectivity for private clouds is deterministic, so deployments primarily rely on Ethernet or other dedicated network services. Some private cloud designs integrate wireless access to selected applications.

The reality is that many enterprises have a mix of applications, requiring the use of multiple public or private clouds. Hybrid implementations that incorporate both public and private cloud functionality are also gaining traction, with advanced deployments integrating resources and cross-domain data sharing.

Ethernet enables cloud connectivity through several service types:

- **Ethernet Private Lines (EPLs)** and **Ethernet Virtual Private Lines (EVPLs)** are the top services for private cloud and inter-data center connectivity. EPLs provide point-to-point connections, while EVPLs also support point-to-multipoint connectivity using EVCs (Ethernet

Virtual Connections). Traffic prioritization is provided through CoS (Class of Service) features.

- **Ethernet DIA** (Dedicated Internet Access) services are used primarily for connectivity to public cloud offerings.
- **E-Access to IP/MPLS VPN** implementations are increasing for hybrid Ethernet/IP VPNs that link to public services or to private clouds.
- **E-LAN** services are used for private cloud connectivity between on-net enterprise sites and data centers. Metro LAN services connect sites within a metro area, and WAN VPLS services support wide area topologies.

Ethernet-based cloud connectivity is also heating up for collocation companies (e.g., Equinix, Telx, etc.). Exchange services offer vendor-neutral connections among cloud providers, content/media providers, network service operators and enterprises. Ethernet simplifies physical connections for exchange participants and enables virtual interconnectivity. These capabilities facilitate new business models that disrupt the economics of traditional wide area networks. Look for exchange ecosystems to expand their cloud offerings during 2013.

Standards for Ethernet-based cloud connectivity continue to advance. The MEF's Carrier Ethernet 2.0 (CE 2.0) initiative provides guidelines for cloud-ready Ethernet services and equipment. Developments are focused on multi-network Interconnectivity, end-to-end SLAs (Service Level Agreements), application-aware QoS (Quality of Service) and dynamic bandwidth provisioning. A new CE 2.0 certification process aims to ensure standards adherence.

There is also strong momentum for Software Defined Networking (SDN). Ethernet providers are evaluating the benefits of SDN to support their IP networks, data centers and cloud services as well as to facilitate the delivery of enhanced capabilities like on-demand service provisioning.

What's really clear about the shift to cloud computing is that network connectivity is essential, and increasingly more complex. Cloud users want high speed, reliable, secure, manageable access to their applications. Monetization opportunities abound for Ethernet providers that can

successfully deliver innovative cloud services and flexible connectivity solutions.

The cloud plays an integral role in how many organizations handle their daily operations, and with remote work and school becoming more prominent than ever, the demand for cloud services is only set to increase.

As more business activities move to the cloud, routing and switching may seem like antiquated components for outdated solutions, but both routers and switches continue to play an integral role in the cloud.

Routing & Switching: A Brief Introduction

Routers are used to tie multiple networks together, such as connecting your internal company network to the internet and are responsible for dictating which devices on your internal network are able to access the internet. The router acts like a dispatcher, directing traffic so that each user can send and receive data as quickly as possible, and it decides which devices need their traffic prioritized. They also play a critical role in safeguarding your network. Routers are responsible for analyzing all the data being sent over your network, dictating how it is packaged and ensuring that data makes it to its destination network.

Switches act like routers on a smaller scale and are used to connect multiple devices on the same network. For example, a company's internal network relies on switches to connect user devices as well as printers, servers, and any IoT devices, creating a shared network of resources and determining how resources are allocated. Switches play a critical role, and their ability to direct internal traffic increases network productivity.

Routers and switches allow workers on your network to access business applications (such as sales tracking applications or financial applications, such as payroll software) whether they are located in the same office or spread out over multiple locations. Keeping everyone connected and ensuring everyone is able to quickly access the business applications, information, and tools they need to do their jobs is critical for productivity. This especially holds true in the era of remote work, when workers are more likely to be accessing business resources from home. Your network needs to be able to support remote workers to help keep your organization running.

The Cloud Still Needs Routing & Switching

Even if you already rely on the cloud or are looking to switch to the cloud, your organization is still going to need routers and switches to direct traffic within your network and between your network and external networks like the internet.

Many cloud-based organizations rely on hybrid cloud and multi-cloud environments, both of which need routers and switches to support connectivity to and from the cloud or clouds, and within each cloud. Hybrid cloud solutions combine private clouds with one or more public cloud services such as AWS.

How Routing & Switching Have Evolved in the Cloud Era

When it comes to technology, innovation is critical for remaining relevant. There have been quite a few innovations regarding virtual routing and switching to support cloud-based networks. Recent innovations have also made it easier than ever to migrate to the cloud and support ongoing operations.

What is a Network Switch?

Top-notch networking gear is a must to keep your organization running smoothly, and a network switch is one of the basic building blocks of your network — simply put, it's a device that connects multiple devices together. Switches allow devices to share and transfer data, enabling communication between devices on the network. Switches work by processing packets of data and routing them to the intended destination(s). In a small business setting, for example, a network switch could be used to connect a computer, printer and server and pass data between all three.

Types of Network Switches

Here are some of the most common types of network switches, with more info on each below:

- KVM Switch
- Managed Switch
- Unmanaged Switch
- Smart Switch

- PoE Switch

KVM Switch

If you guessed that "KVM" stands for "keyboard, video and mouse," you would be correct. This type of switch is used to connect a keyboard, mouse or monitor to multiple computers. These switches are often used to control groups of servers while saving desktop space by eliminating cables.

A KVM switch is an ideal interface for a single user that needs to control the functions of multiple computers from a single console. These devices can often be programmed with keyboard hotkeys that let you easily switch between PCs. With the addition of a KVM extender, the reach of the switch can be extended several hundred feet by transmitting DVI, VGA or HDMI video signals. This configuration allows for local and remote access to the machines. A complete KVM solution lets you easily centralize server maintenance and management.

Managed Switch

A managed switch is exactly what it sounds like—a switch that requires some oversight by a network administrator. This type of switch gives you total control over the traffic accessing your network while allowing you to custom-configure each Ethernet port so you get maximum efficiency over data transfers on the network. Administrators can tweak these devices for optimal data rate as new devices and users are added to the network through commands such as bandwidth rate limiting and port mirroring. Managed switches are also typically the best network switches to support the Gigabit standard of Ethernet rather than traditional Fast Ethernet.

Many administrators use managed switches to create virtual local area networks (VLANs), which allow you to further segment your network and control the traffic burden for each type of connected device. Another benefit of a managed switch setup is that the majority of managed switches are designed with Spanning Tree Protocol (STP). This enables administrators to perform quality of service (QoS) duties and access the switch remotely to make adjustments without having to be in the same physical location as the switch. Managed switches are often higher in cost than their unmanaged counterparts, but the payoff is that you have the freedom to create a network that runs at peak efficiency customized to the specifications of the unique devices on it.

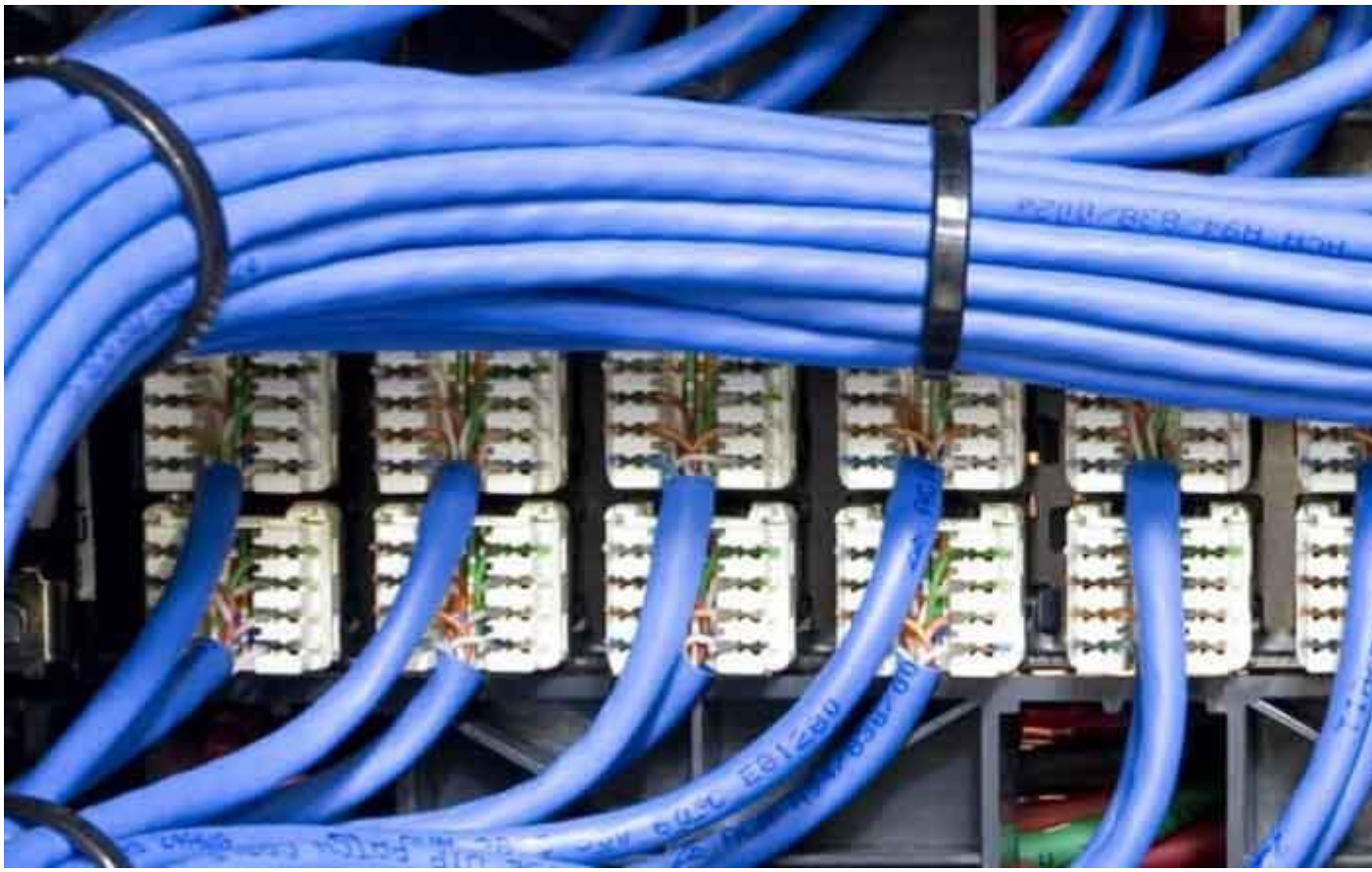
Unmanaged Switch

Unmanaged switches are generally made as plug-and-play devices and require little to no special installation beyond an Ethernet cable. The setup of this type of switch relies

on auto-negotiation between Ethernet devices to enable communication between them. The switch will automatically determine the best data rate to use, switching between full-duplex mode (where data is received or transmitted in two directions at the same time) or half-duplex mode (where data is received or transmitted two ways but only one direction at a time).

While some unmanaged switches may be accessed remotely, most will require the admin to physically make broad changes when setting up the switch. If you want a switch that will perform the basic functions of network efficiency without the need for customization, unmanaged may be the best the type of network switch for you.

Debating between a managed and unmanaged switch purchase? Be sure to read our detailed comparison of managed vs. unmanaged switches.



Smart Switch

Another popular type of switch in networking is the smart switch, also referred to as an intelligent switch. These devices are a type of managed switch with only a select number of options for management. Rather than providing the full management functionality of a managed switch, a smart switch may only provide functionality to configure a handful of settings, like VLANs or duplex modes.

If your network will not require a full set of customizations, a smart switch can be a good option. These devices are often more affordable than full managed switches while still offering more customization options compared to unmanaged switches.

PoE Switch/Injector

PoE stands for power over Ethernet. A PoE switch distributes power over the network to different devices. This means any device on the network, from PCs to IP cameras and smart lighting systems, can function without the need to be near an AC access point or router, because the PoE switch sends both data and power to the connected devices.

While a PoE switch creates a PoE network that can support both Ethernet and PoE-enabled devices, a PoE injector takes it up a level. The injector takes a device with both PoE and non-PoE switches and uses those to create access points as needed for devices on the network using a Cat 5 cable or better to transmit the necessary electricity to function over signal wires. By utilizing the power of a PoE injector when needed, you can create a work or home network that runs efficiently without the need to have additional power supplies installed for various devices. However, not all devices are compatible with every PoE switch or injector. Be sure to check if your PoE switch is compliant with the 802.3af/at standard and if the device you want to connect can support that.

Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

Features of Docker

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

Components of Docker

Docker has the following components

- **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
- **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
- **Docker for Windows** − It allows one to run Docker containers on the Windows OS.
- **Docker Engine** − It is used for building Docker images and creating Docker containers.
- **Docker Hub** − This is the registry which is used to host various Docker images.
- **Docker Compose** − This is used to define applications using multiple Docker containers.

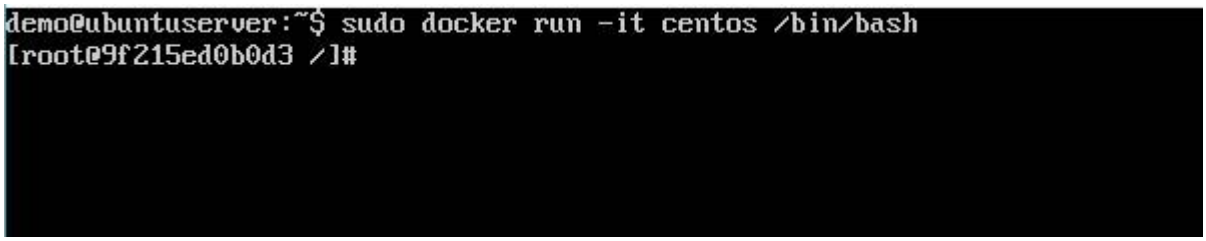
Containers are instances of Docker images that can be run using the Docker run command. The basic purpose of Docker is to run containers. Let's discuss how to work with containers.

Running a Container

Running of containers is managed with the Docker **run** command. To run a container in an interactive mode, first launch the Docker container.

```
sudo docker run –it centos /bin/bash
```

Then hit Crtl+p and you will return to your OS shell.



You will then be running in the instance of the CentOS system on the Ubuntu server.

Listing of Containers

One can list all of the containers on the machine via the **docker ps** command. This command is used to return the currently running containers.

```
docker ps
```

Syntax

```
docker ps
```

Options

None

Return Value

The output will show the currently running containers.

Example

```
sudo docker ps
```

Output

When we run the above command, it will produce the following result −

Let's see some more variations of the **docker ps** command.

docker ps -a

This command is used to list all of the containers on the system

Syntax

docker ps -a

Options

- **—a** − It tells the **docker ps** command to list all of the containers on the system.

Return Value

The output will show all containers.

Example

```
sudo docker ps -a
```

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
   STATUS                      PORTS
     NAMES
9f215ed0b0d3        centos:latest       "/bin/bash"              4 minutes ago
   Up 4 minutes
     cocky_colden
e5a02936065a        centos:latest       "/bin/bash"              39 minutes ago
   Exited (0) 39 minutes ago
     ecstatic_hodgkin
9b286dd1f16a        jenkins:latest      "/bin/tini -- /usr/l   18 hours ago
   Exited (0) About an hour ago   0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000
cp   jolly_wright
3646aa260a2d        jenkins:latest      "/bin/tini -- /usr/l   9 days ago
   Exited (0) 9 days ago          0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000
cp   reverent_morse
demo@ubuntuserver:~$ _
```

docker history

With this command, you can see all the commands that were run with an image via a container.

Syntax

docker history ImageID

Options

- **ImageID** − This is the Image ID for which you want to see all the commands that were run against it.

Return Value

The output will show all the commands run against that image.

Example

```
sudo docker history centos
```

The above command will show all the commands that were run against the **centos** image.

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
jenkins             latest              998d1854867e        2 weeks ago
714.1 MB
centos              latest              97cad5e16cb6        4 weeks ago
196.5 MB
demo@ubuntuserver:~$ sudo docker history centos
IMAGE               CREATED             CREATED BY
        SIZE
97cad5e16cb6        4 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]
        0 B
05fe84bf6d3f        4 weeks ago         /bin/sh -c #(nop)  LABEL name=CentOS B
e Ima   0 B
af0819ed1fac        4 weeks ago         /bin/sh -c #(nop) ADD file:54df3580ac9
66389   196.5 MB
3690474eb5b4        3 months ago        /bin/sh -c #(nop)  MAINTAINER https://
thub.   0 B
demo@ubuntuserver:~$ _
```

In this chapter, we will explore in detail what we can do with containers.

docker top

With this command, you can see the top processes within a container.

Syntax

```
docker top ContainerID
```

Options

- **ContainerID** − This is the Container ID for which you want to see the top processes.

Return Value

The output will show the top-level processes within a container.

Example

```
sudo docker top 9f215ed0b0d3
```

The above command will show the top-level processes within a container.

Output

When we run the above command, it will produce the following result −

docker stop

This command is used to stop a running container.

Syntax

docker stop ContainerID

Options

- **ContainerID** − This is the Container ID which needs to be stopped.

Return Value

The output will give the ID of the stopped container.

Example

```
sudo docker stop 9f215ed0b0d3
```

The above command will stop the Docker container **9f215ed0b0d3**.

Output

When we run the above command, it will produce the following result −

docker rm

This command is used to delete a container.

Syntax

docker rm ContainerID

Options

- **ContainerID** − This is the Container ID which needs to be removed.

Return Value

The output will give the ID of the removed container.

Example

```
sudo docker rm 9f215ed0b0d3
```

The above command will remove the Docker container **9f215ed0b0d3**.

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
9f215ed0b0d3        centos:latest       "/bin/bash"         22 minutes ago
Up 22 minutes                           cocky_colden
demo@ubuntuserver:~$ sudo docker stop 9f215ed0b0d3
9f215ed0b0d3
demo@ubuntuserver:~$ sudo docker rm 9f215ed0b0d3
9f215ed0b0d3
demo@ubuntuserver:~$ _
```

docker stats

This command is used to provide the statistics of a running container.

Syntax

docker stats ContainerID

Options

- **ContainerID** − This is the Container ID for which the stats need to be provided.

Return Value

The output will show the CPU and Memory utilization of the Container.

Example

```
sudo docker stats 9f215ed0b0d3
```

The above command will provide CPU and memory utilization of the Container **9f215ed0b0d3**.

Output

When we run the above command, it will produce the following result −

```
CONTAINER           CPU %               MEM USAGE/LIMIT     MEM %
NET I/O
07b0b6f434fe        0.00%               416 KiB/1.416 GiB   0.03%
648 B/648 B
```

docker attach

This command is used to attach to a running container.

Syntax

docker attach ContainerID

Options

- **ContainerID** − This is the Container ID to which you need to attach.

Return Value

None

Example

```
sudo docker attach 07b0b6f434fe
```

The above command will attach to the Docker container **07b0b6f434fe**.

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
07b0b6f434fe        centos:latest       "/bin/bash"         3 minutes ago
Up 3 minutes                            cocky_pare
demo@ubuntuserver:~$ sudo docker attach 07b0b6f434fe

[root@07b0b6f434fe /]# _
```

Once you have attached to the Docker container, you can run the above command to see the process utilization in that Docker container.

```
top - 15:24:06 up  2:06,  0 users,  load average: 0.00, 0.01, 0.02
Tasks:   2 total,   1 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
KiB Mem :  1484856 total,  1057152 free,    52368 used,   375336 buff/cache
KiB Swap:  1519612 total,  1519612 free,        0 used.  1403868 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0   11784   2992   2644 S   0.0   0.2   0:00.01 bash
   15 root      20   0   51864   3772   3272 R   0.0   0.3   0:00.00 top
```

docker pause

This command is used to pause the processes in a running container.

Syntax

docker pause ContainerID

Options

- **ContainerID** − This is the Container ID to which you need to pause the processes in the container.

Return Value

The ContainerID of the paused container.

Example

```
sudo docker pause 07b0b6f434fe
```

The above command will pause the processes in a running container **07b0b6f434fe**.

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker ps
[sudo] password for demo:
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
07b0b6f434fe        centos:latest       "/bin/bash"         18 minutes ago
Up 18 minutes                           cocky_pare
demo@ubuntuserver:~$ sudo docker pause 07b0b6f434fe
07b0b6f434fe
demo@ubuntuserver:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                  PORTS               NAMES
07b0b6f434fe        centos:latest       "/bin/bash"         19 minutes ago
Up 19 minutes (Paused)                      cocky_pare
demo@ubuntuserver:~$ _
```

docker unpause

This command is used to **unpause** the processes in a running container.

Syntax

docker unpause ContainerID

Options

- **ContainerID** − This is the Container ID to which you need to unpause the processes in the container.

Return Value

The ContainerID of the running container.

Example

```
sudo docker unpause 07b0b6f434fe
```

The above command will unpause the processes in a running container: 07b0b6f434fe

Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker unpause 07b0b6f434fe
07b0b6f434fe
demo@ubuntuserver:~$
```

docker kill

This command is used to kill the processes in a running container.

Syntax

docker kill ContainerID

Options

- **ContainerID** − This is the Container ID to which you need to kill the processes in the container.

Return Value

The ContainerID of the running container.

Example

```
sudo docker kill 07b0b6f434fe
```

The above command will kill the processes in the running container **07b0b6f434fe**.

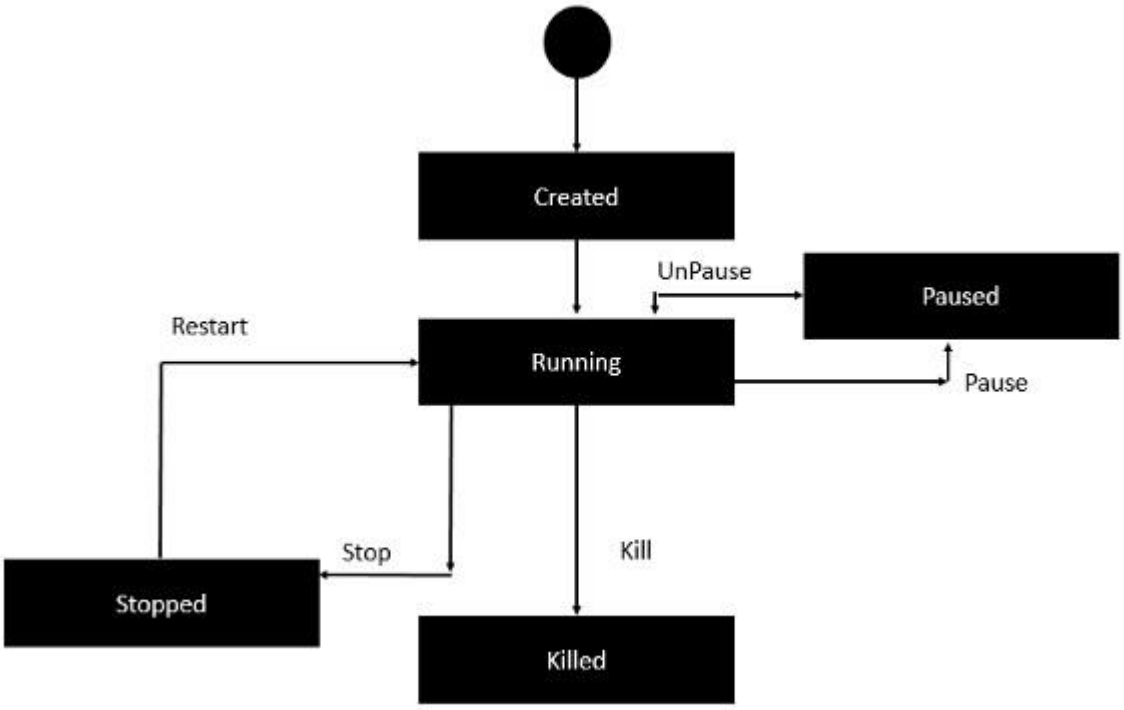
Output

When we run the above command, it will produce the following result −

```
demo@ubuntuserver:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
07b0b6f434fe        centos:latest       "/bin/bash"         23 minutes ago
Up 23 minutes                           cocky_pare
demo@ubuntuserver:~$ sudo docker kill 07b0b6f434fe
07b0b6f434fe
demo@ubuntuserver:~$
```

Docker – Container Lifecycle

The following illustration explains the entire lifecycle of a Docker container.



- Initially, the Docker container will be in the **created** state.
- Then the Docker container goes into the running state when the Docker **run** command is used.
- The Docker **kill** command is used to kill an existing Docker container.
- The Docker **pause** command is used to pause an existing Docker container.
- The Docker **stop** command is used to pause an existing Docker container.
- The Docker **run** command is used to put a container back from a **stopped** state to a **running** state.

Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF). This is also known as the enhanced version of Borg which was developed at Google to manage both long running processes and batch jobs, which was earlier handled by separate systems.

Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure.

Features of Kubernetes

Following are some of the important features of Kubernetes.

- Continues development, integration and deployment
- Containerized infrastructure
- Application-centric management
- Auto-scalable infrastructure
- Environment consistency across development testing and production
- Loosely coupled infrastructure, where each component can act as a separate unit

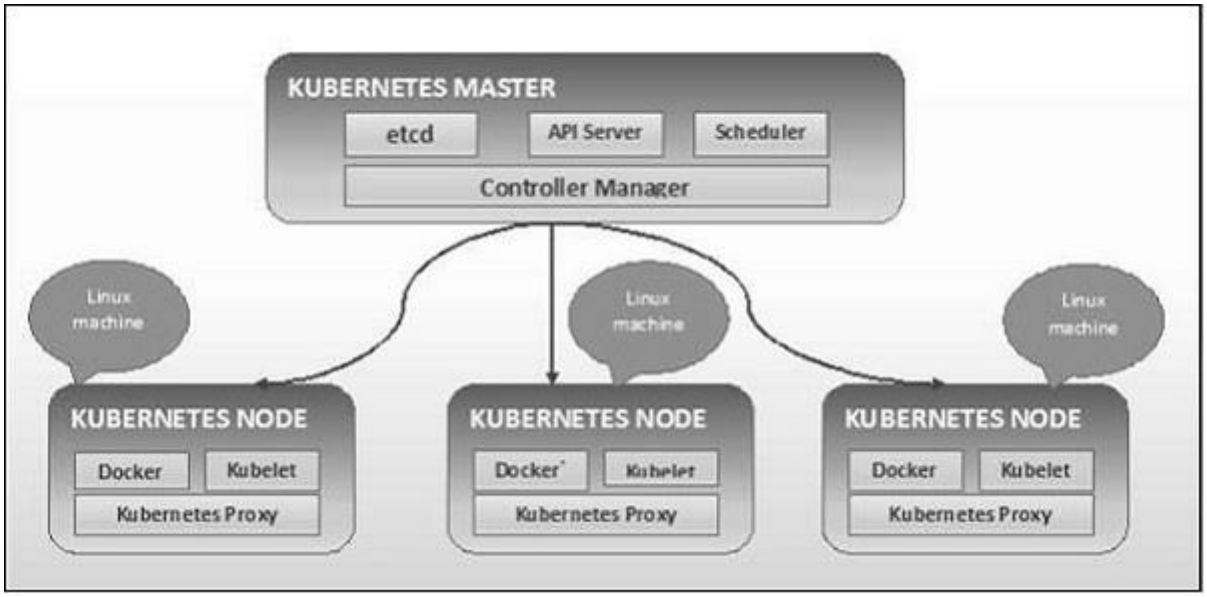
- Higher density of resource utilization
- Predictable infrastructure which is going to be created

One of the key components of Kubernetes is, it can run application on clusters of physical and virtual machine infrastructure. It also has the capability to run applications on cloud. **It helps in moving from host-centric infrastructure to container-centric infrastructure.**

In this chapter, we will discuss the basic architecture of Kubernetes.

Kubernetes - Cluster Architecture

As seen in the following diagram, Kubernetes follows client-server architecture. Wherein, we have master installed on one machine and the node on separate Linux machines.



The key components of master and node are defined in the following section.

Kubernetes - Master Machine Components

Following are the components of Kubernetes Master Machine.

etcd

It stores the configuration information which can be used by each of the nodes in the cluster. It is a high availability key value store that can be distributed among multiple nodes. It is accessible only by Kubernetes API server as it may have some sensitive information. It is a distributed key value Store which is accessible to all.

API Server

Kubernetes is an API server which provides all the operation on cluster using the API. API server implements an interface, which means different tools and libraries can readily

communicate with it. **Kubeconfig** is a package along with the server side tools that can be used for communication. It exposes Kubernetes API.

Controller Manager

This component is responsible for most of the collectors that regulates the state of cluster and performs a task. In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server. It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state. The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

Scheduler

This is one of the key components of Kubernetes master. It is a service in master responsible for distributing the workload. It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload. In other words, this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.

Kubernetes - Node Components

Following are the key components of Node server which are necessary to communicate with Kubernetes master.

Docker

The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment.

Kubelet Service

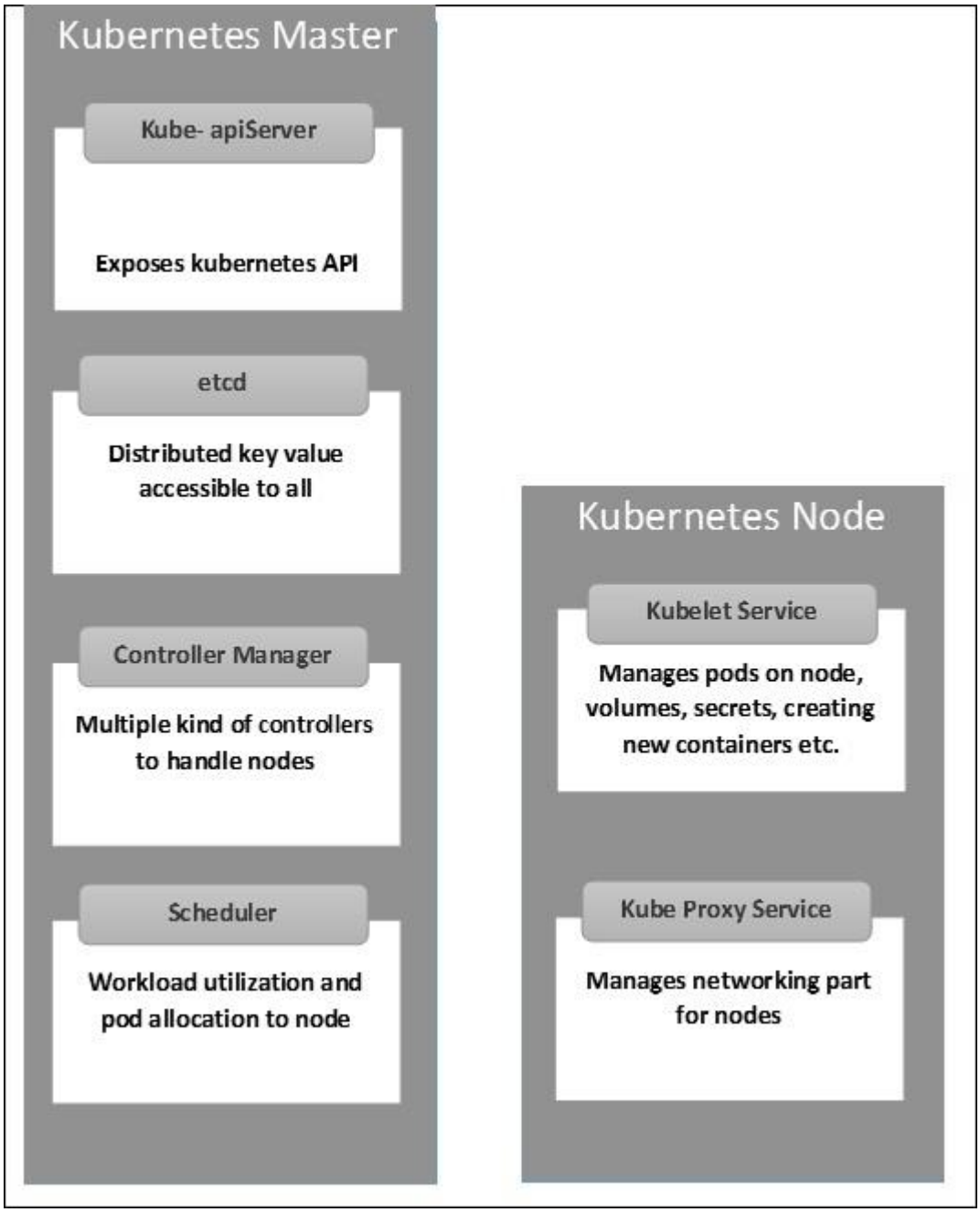
This is a small service in each node responsible for relaying information to and from control plane service. It interacts with **etcd** store to read configuration details and wright values. This communicates with the master component to receive commands and work. The **kubelet** process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

Kubernetes Proxy Service

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers' health checkup, etc.

Kubernetes - Master and Node Structure

The following illustrations show the structure of Kubernetes Master and Node.



It is important to set up the Virtual Datacenter (vDC) before setting up Kubernetes. This can be considered as a set of machines where they can communicate with each other via the network. For hands-on approach, you can set up vDC on **PROFITBRICKS** if you do not have a physical or cloud infrastructure set up.

Once the IaaS setup on any cloud is complete, you need to configure the **Master** and the **Node**.

Note − The setup is shown for Ubuntu machines. The same can be set up on other Linux machines as well.

Prerequisites

Installing Docker − Docker is required on all the instances of Kubernetes. Following are the steps to install the Docker.

Step 1 − Log on to the machine with the root user account.

Step 2 − Update the package information. Make sure that the apt package is working.

Step 3 − Run the following commands.

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
```

Step 4 − Add the new GPG key.

```
$ sudo apt-key adv \
   --keyserver hkp://ha.pool.sks-keyservers.net:80 \
   --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
$ echo "deb https://apt.dockerproject.org/repo ubuntu-trusty main" | sudo tee
/etc/apt/sources.list.d/docker.list
```

Step 5 − Update the API package image.

```
$ sudo apt-get update
```

Once all the above tasks are complete, you can start with the actual installation of the Docker engine. However, before this you need to verify that the kernel version you are using is correct.

Install Docker Engine

Run the following commands to install the Docker engine.

Step 1 − Logon to the machine.

Step 2 − Update the package index.

```
$ sudo apt-get update
```

Step 3 − Install the Docker Engine using the following command.

```
$ sudo apt-get install docker-engine
```

Step 4 − Start the Docker daemon.

```
$ sudo apt-get install docker-engine
```

Step 5 − To very if the Docker is installed, use the following command.

```
$ sudo docker run hello-world
```

Install etcd 2.0

This needs to be installed on Kubernetes Master Machine. In order to install it, run the following commands.

```
$ curl -L https://github.com/coreos/etcd/releases/download/v2.0.0/etcd
-v2.0.0-linux-amd64.tar.gz -o etcd-v2.0.0-linux-amd64.tar.gz ->1
$ tar xzvf etcd-v2.0.0-linux-amd64.tar.gz ------>2
$ cd etcd-v2.0.0-linux-amd64 ------------>3
$ mkdir /opt/bin ------------>4
$ cp etcd* /opt/bin ----------->5
```

In the above set of command −

- First, we download the **etcd**. Save this with specified name.
- Then, we have to un-tar the tar package.
- We make a dir. inside the /opt named bin.
- Copy the extracted file to the target location.

Now we are ready to build Kubernetes. We need to install Kubernetes on all the machines on the cluster.

```
$ git clone https://github.com/GoogleCloudPlatform/kubernetes.git
$ cd kubernetes
$ make release
```

The above command will create a _**output** dir in the root of the kubernetes folder. Next, we can extract the directory into any of the directory of our choice /opt/bin, etc.

Next, comes the networking part wherein we need to actually start with the setup of Kubernetes master and node. In order to do this, we will make an entry in the host file which can be done on the node machine.

```
$ echo "<IP address of master machine> kube-master
< IP address of Node Machine>" >> /etc/hosts
```

Following will be the output of the above command.

```
root@boot2docker:/etc# cat /etc/hosts
127.0.0.1 boot2docker localhost localhost.local

# The following lines are desirable for IPv6 capable hosts
# (added automatically by netbase upgrade)

::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

10.11.50.12 kube-master
10.11.50.11  kube-minion
```

Now, we will start with the actual configuration on Kubernetes Master.

First, we will start copying all the configuration files to their correct location.

$ cp <Current dir. location>/kube-apiserver /opt/bin/
$ cp <Current dir. location>/kube-controller-manager /opt/bin/
$ cp <Current dir. location>/kube-kube-scheduler /opt/bin/
$ cp <Current dir. location>/kubecfg /opt/bin/
$ cp <Current dir. location>/kubectl /opt/bin/
$ cp <Current dir. location>/kubernetes /opt/bin/

The above command will copy all the configuration files to the required location. Now we will come back to the same directory where we have built the Kubernetes folder.

$ cp kubernetes/cluster/ubuntu/init_conf/kube-apiserver.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-controller-manager.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-kube-scheduler.conf /etc/init/

$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-apiserver /etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-controller-manager /etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-kube-scheduler /etc/init.d/

$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kube-proxy /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/

The next step is to update the copied configuration file under /etc. dir.

Configure etcd on master using the following command.

$ ETCD_OPTS = "-listen-client-urls = http://kube-master:4001"

Configure kube-apiserver

For this on the master, we need to edit the **/etc/default/kube-apiserver** file which we copied earlier.

$ KUBE_APISERVER_OPTS = "--address = 0.0.0.0 \
--port = 8080 \
--etcd_servers = <The path that is configured in ETCD_OPTS> \
--portal_net = 11.1.1.0/24 \
--allow_privileged = false \
--kubelet_port = < Port you want to configure> \
--v = 0"

Configure the kube Controller Manager

We need to add the following content in **/etc/default/kube-controller-manager**.

$ KUBE_CONTROLLER_MANAGER_OPTS = "--address = 0.0.0.0 \
--master = 127.0.0.1:8080 \
--machines = kube-minion \ -----> #this is the kubernatics node
--v = 0

Next, configure the kube scheduler in the corresponding file.

```
$ KUBE_SCHEDULER_OPTS = "--address = 0.0.0.0 \
--master = 127.0.0.1:8080 \
--v = 0"
```

Once all the above tasks are complete, we are good to go ahead by bring up the Kubernetes Master. In order to do this, we will restart the Docker.

```
$ service docker restart
```

Kubernetes Node Configuration

Kubernetes node will run two services the **kubelet and the kube-proxy**. Before moving ahead, we need to copy the binaries we downloaded to their required folders where we want to configure the kubernetes node.

Use the same method of copying the files that we did for kubernetes master. As it will only run the kubelet and the kube-proxy, we will configure them.

```
$ cp <Path of the extracted file>/kubelet /opt/bin/
$ cp <Path of the extracted file>/kube-proxy /opt/bin/
$ cp <Path of the extracted file>/kubecfg /opt/bin/
$ cp <Path of the extracted file>/kubectl /opt/bin/
$ cp <Path of the extracted file>/kubernetes /opt/bin/
```

Now, we will copy the content to the appropriate dir.

```
$ cp kubernetes/cluster/ubuntu/init_conf/kubelet.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-proxy.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kubelet /etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-proxy /etc/init.d/
$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kube-proxy /etc/default/
```

We will configure the **kubelet** and **kube-proxy conf** files.

We will configure the **/etc/init/kubelet.conf**.

```
$ KUBELET_OPTS = "--address = 0.0.0.0 \
--port = 10250 \
--hostname_override = kube-minion \
--etcd_servers = http://kube-master:4001 \
--enable_server = true
--v = 0"
/
```

For kube-proxy, we will configure using the following command.

```
$ KUBE_PROXY_OPTS = "--etcd_servers = http://kube-master:4001 \
--v = 0"
/etc/init/kube-proxy.conf
```

Finally, we will restart the Docker service.

```
$ service docker restart
```

Now we are done with the configuration. You can check by running the following commands.

```
$ /opt/bin/kubectl get minions
```

Kubernetes - Images

Kubernetes (Docker) images are the key building blocks of Containerized Infrastructure. As of now, we are only supporting Kubernetes to support Docker images. Each container in a pod has its Docker image running inside it.

When we are configuring a pod, the image property in the configuration file has the same syntax as the Docker command does. The configuration file has a field to define the image name, which we are planning to pull from the registry.

Following is the common configuration structure which will pull image from Docker registry and deploy in to Kubernetes container.

```
apiVersion: v1
kind: pod
metadata:
  name: Tesing_for_Image_pull ----------> 1
  spec:
    containers:
      - name: neo4j-server ----------------------> 2
      image: <Name of the Docker image>----------> 3
      imagePullPolicy: Always ------------>4
      command: ["echo", "SUCCESS"] ------------------->
```

In the above code, we have defined −

- **name: Tesing_for_Image_pull** − This name is given to identify and check what is the name of the container that would get created after pulling the images from Docker registry.
- **name: neo4j-server** − This is the name given to the container that we are trying to create. Like we have given neo4j-server.
- **image: <Name of the Docker image>** − This is the name of the image which we are trying to pull from the Docker or internal registry of images. We need to define a complete registry path along with the image name that we are trying to pull.
- **imagePullPolicy** − Always - This image pull policy defines that whenever we run this file to create the container, it will pull the same name again.
- **command: [“echo”, “SUCCESS”]** − With this, when we create the container and if everything goes fine, it will display a message when we will access the container.

In order to pull the image and create a container, we will run the following command.

```
$ kubectl create –f Tesing_for_Image_pull
```

Once we fetch the log, we will get the output as successful.

```
$ kubectl log Tesing_for_Image_pull
```

The above command will produce an output of success or we will get an output as failure.