



BITS Pilani

Cloud Computing

CS G527

Dr. D.V.N. Siva Kumar
CSIS Department, BITS Pilani, Hyderabad Campus



What is Datacenter?

Who need Datacenters?

What could be total cost of setting up a typical Datacenter (including its maintenance for a period of 5 years)?

What is Datacenter?

Ans: It is a building, dedicated space within a building, or a group of buildings used to house computer systems and associated components, such as telecommunications and storage systems..

In other words, the Data centers are just centralized locations where computing and networking equipment is used for the purpose of collecting, storing, processing, distributing or allowing access to mass amounts of data.

Things involved in setting up a Data Center and Its maintenance

1. **Facility** –Location and the usable space

2. **Support Infrastructure:**

UPS - Uninterruptible power sources

Physical security systems – the controlling of entrance and exits of the facility involves biometrics and video surveillance systems

3. **IT equipment** – This is the core of the data center and contains IT operations and storage equipment's which includes **servers, racks, cables, storage devices** and to **Maintain a vigilance** on these crucial devices with firewalls and security devices.

4. **Operation Staff**

5. **Licensing & Support:** The cost involved in buying the OS, software etc., and the efforts involved in getting support from the vendor when things screw up.

Reference Link : <https://www.quora.com/What-is-required-to-set-up-a-data-center>

Who need Datacenters?

Ans:

- **Any entity that generates or uses data has the need for data centers** including government agencies, educational bodies, telecommunication companies, financial institutions, retailers of all sizes, and the social networking services such as Google and Facebook.
- Some build and maintain them in-house and some rent servers at co-location facilities.
- Some even use public cloud-based services too.

Note: Lack of fast and reliable access to data would mean the inability to provide vital services or loss of customer satisfaction and revenue.

Cost Comparison of On-premises Servers versus Cloud

What could be the total cost of setting up a datacenter and its maintenance?

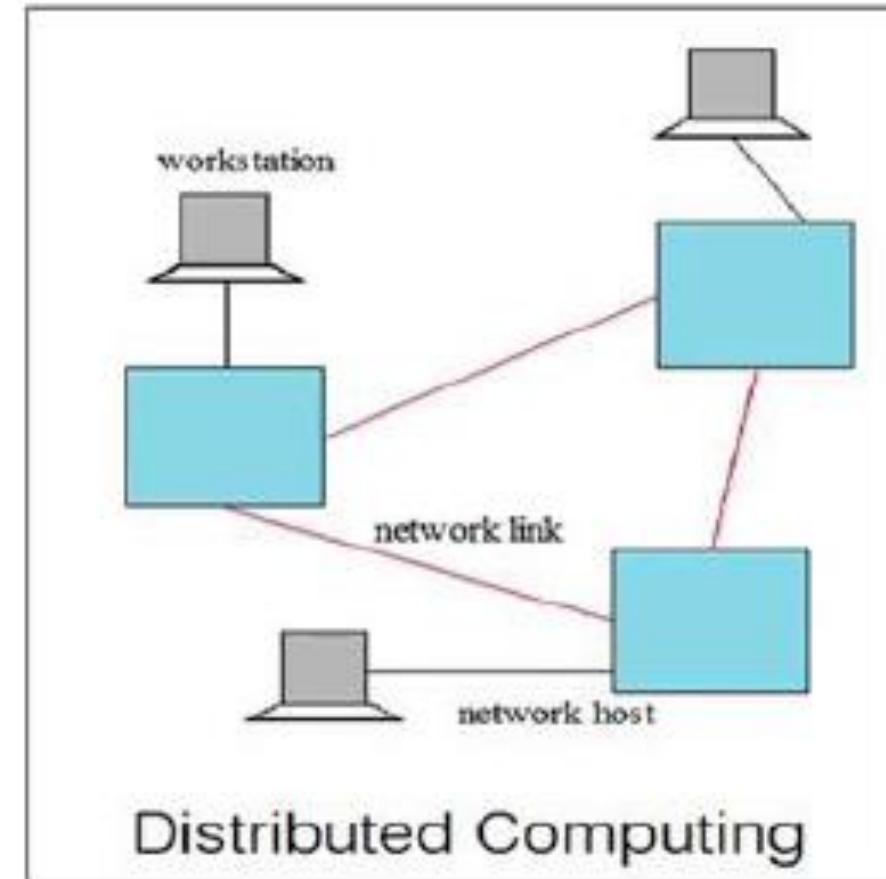
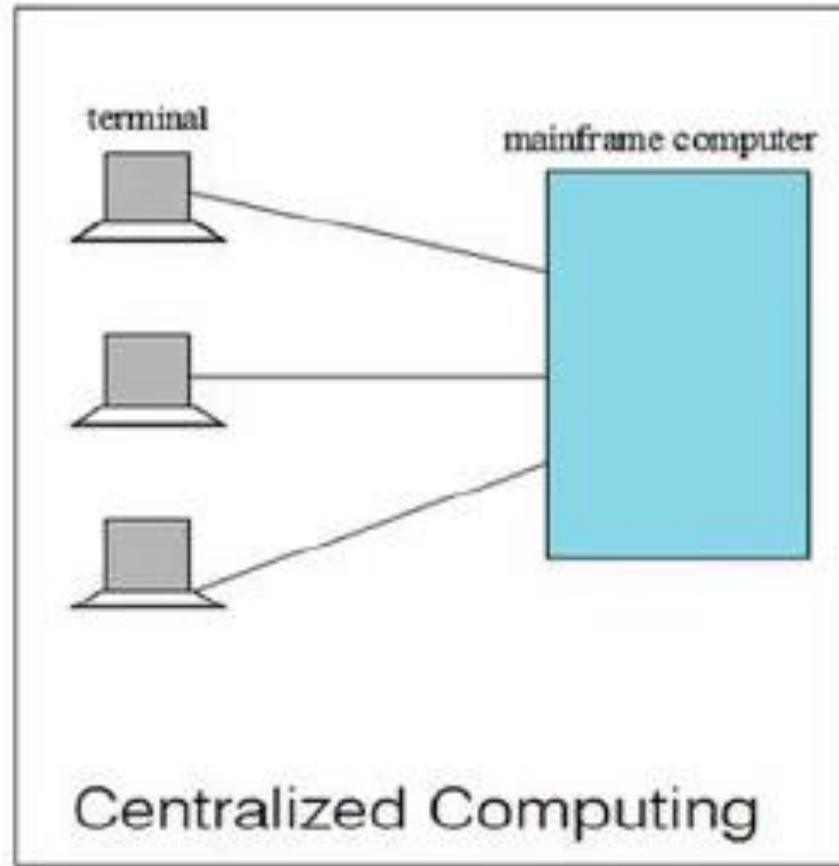
	On-premises	Cloud	Savings \$	Saving %
Year 1	39,347.18 \$	3,766.80 \$	35,580.38 \$	90%
Year 2	9,063.19 \$	3,766.80 \$	5,296.39 \$	58%
Year 3	9,063.19 \$	3,766.80 \$	5,296.39 \$	58%
Year 4	9,063.19 \$	3,766.80 \$	5,296.39 \$	58%
Year 5	39,347.18 \$	3,766.80 \$	35,580.38 \$	90%
Year 6	9,063.19 \$	3,766.80 \$	5,296.39 \$	58%
Year 7	9,063.19 \$	3,766.80 \$	5,296.39 \$	58%
Total:	124,010.31 \$	26,367.60 \$	97,642.71 \$	79%

URL reference: <https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premise/>

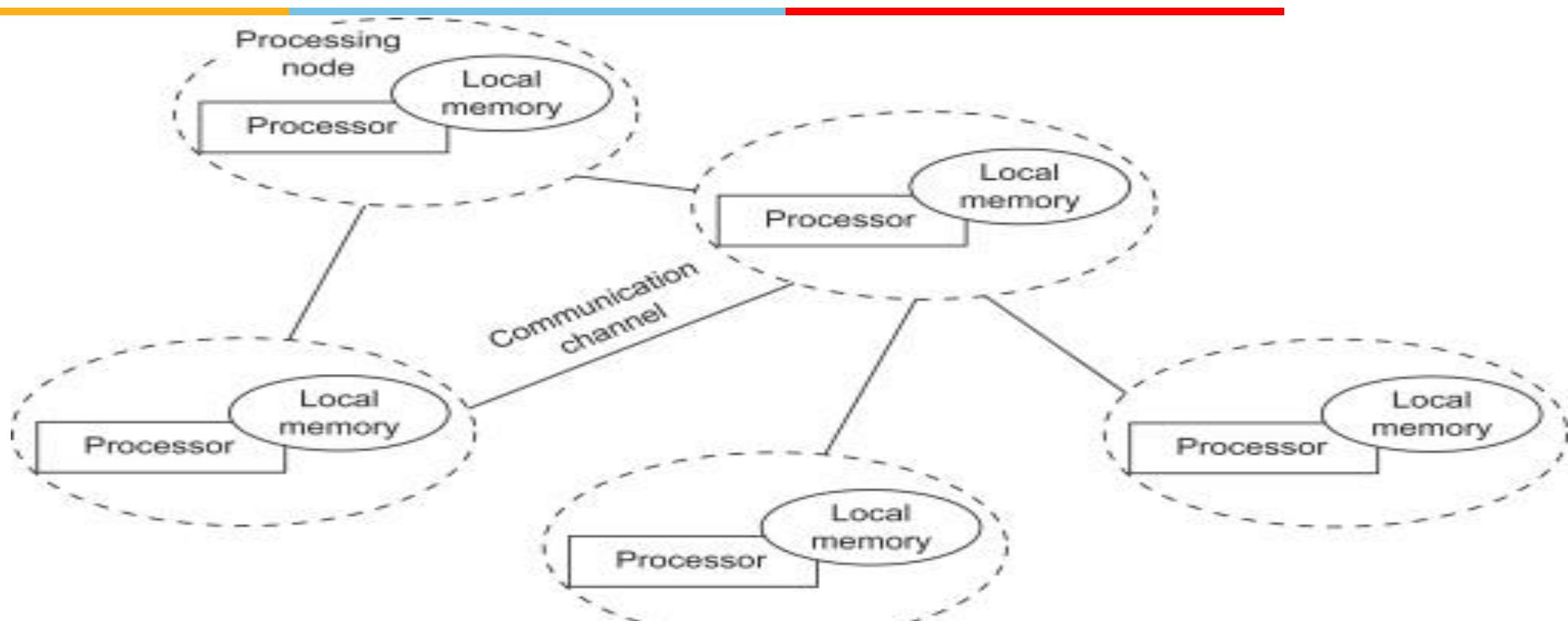
Computing Trends

- Distributed Computing
- Grid Computing
- Cluster Computing
- Utility Computing
- **Cloud Computing**

Centralized Computing Versus Distributed Computing



Distributed System or Computing



Def: A distributed system is a **collection of independent components located on different machines** that share messages with each other in order to achieve common goals.

-Distributed computing (DC) is computing over distributed autonomous computers that communicate only over a network. DC is aimed to improve efficiency and performance.

Examples of Distributed Systems are ATMs(Bank Machines), Internet, etc.

Ref: <https://www.sciencedirect.com/topics/computer-science/distributed-computing>

Why Distributed Computing?

- Nature of Application
- Performance
 - Computing Intensive: The task could consume a lot of time in computing
 - Data Intensive: The task that deals with large size of datasets.
- Robustness:
 - No Single Point of Failure
 - Other nodes can execute the same task executed on the failed node.

Advantages and Disadvantages of Distributed Computing

Major benefits include:

- **Unlimited Horizontal Scaling** - machines can be added whenever required.
- **Low Latency** - having machines that are geographically located closer to users, it will reduce the time it takes to serve users.
- **Fault Tolerance** - if one server or data centre goes down, others could still serve the users of the service.

Disadvantages:

- Data Integration & Consistency
- Network and Communication Failure
- Management Overhead

Computers in a Distributed Computing

- **Workstations:** Computers used by end users to perform computing.
- **Servers:** Computers which provide resources and services
- **Personal Assistance Devices:** Handheld devices connected to the system via a wireless network.

Grid Computing

Def(Grid):

A grid can be defined as a large-scale geographically **distributed** hardware and software infrastructure composed of **heterogeneous networked resources** owned and shared by multiple administrative organizations which are coordinated to provide **transparent, dependable, pervasive and consistent** computing support to a wide range of applications.

- The grid size may vary from small to large enterprises network.
- These applications can perform either distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing.

Ref: https://www.gsic.uva.es/uploaded_files/BoteACG03.pdf

Grid Computing (Cont...)

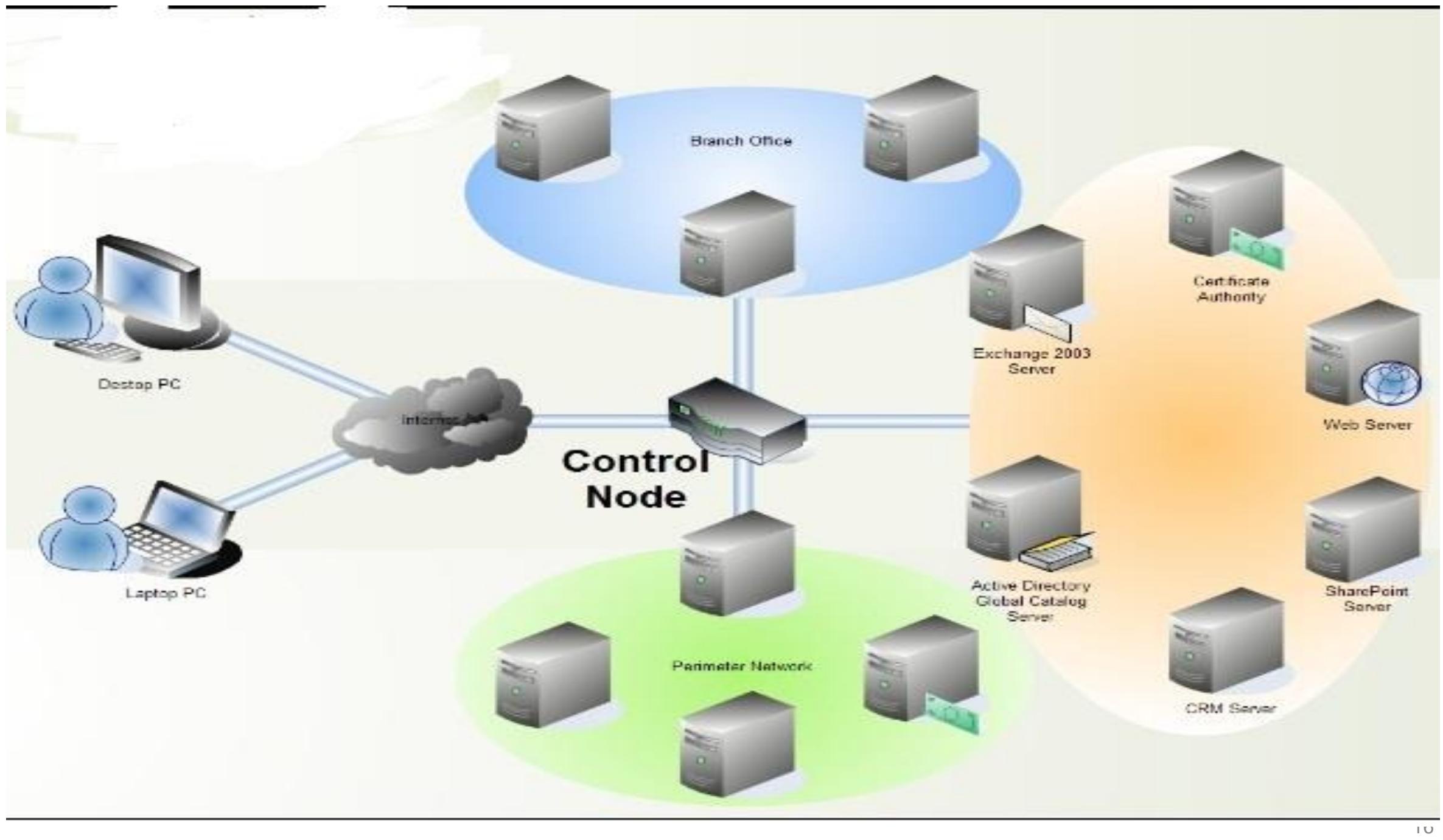
Grid computing is a group of computers physically connected (over a network or with [Internet](#)) to perform a dedicated tasks together, such as analyzing e-commerce data and solve a complex problem

- All machines on that network work under the same protocol to act like a virtual supercomputer. The task that they work on may include analyzing huge datasets or simulating situations which require high computing power.
- Computers on the network contribute resources like processing power and storage capacity to the network.

Note: Grid Computing is a subset of distributed computing, where a virtual super computer comprises of machines on a network connected by some bus, mostly Ethernet or sometimes the Internet.

Grid Computing (Cont...)

- Users (or client applications) gain access to computing resources (processors, storage, data, applications and so on) as needed without any information about the location of these resources and the underlying hardware and operating systems.
- The Grid links together computing resources (PCs, Workstations, Servers, Storage elements and provides a mechanism to access these.



Need of Grid Computing

The basic idea of Grid Computing is to utilize the idle CPU cycles and storage of million of computer systems across a worldwide network function as a flexible, pervasive, and inexpensive accessible pool that could be **harnessed by anyone who needs it**, similar to the way power companies and their users share the electrical grid.

- Exploiting underutilized resources.
- Today's Science/Research is based on computations, data analysis, data visualization and collaborations where grid computing could be helpful.
- Scientific and Engineering problems are becoming more complex and users prefer more accurate, precise solutions to their problems in the shortest possible time.

Example applications of Grid computing are in various domains: Weather forecast applications, Protein Analysis, Detection and modelling natural disasters, etc.

Types of Grids

Computational Grid: It provide secure access to huge pool of shared processing power suitable for high throughput applications and computation intensive computing.

Data Grid: It provides an infrastructure to support data storage, data discovery, data handling, data publication, data manipulation of large volumes of data actually stored in various heterogeneous databases and file systems.

Collaborative Grid: It is the grid which solves collaborative problems.

Advantages and Disadvantages of Grid Computing

Advantages:

- It can solve more complex problems in a very short span of time.
- It can easily combine with other organizations.
- It can make better use of existing hardware.
- Scalability

Disadvantages:

- Challenges with sharing resources (especially across different admin domains)
- It is very non interactive.

Cluster Computing

Clustering refers to establishing connectivity among two or more servers in order to make it work like one.

- **Cluster computing** or *High-Performance computing* frameworks is a form of computing in which bunch of computers (often called nodes) that are connected through a LAN (local area network) so that, they behave like a single machine.
- **A computer cluster is a single logical unit consisting of multiple computers that are linked through a LAN.** The networked computers essentially act as a single, much more powerful machine.
- A computer cluster provides much faster processing speed, larger storage capacity, better data integrity, superior reliability and wider availability of resources.
- The connected computers execute operations all together thus creating the impression like a single system (virtual machine).

Ref: <https://www.watelectronics.com/cluster-computing-architecture-its-types/>

Need for Cluster Computing

- It resolves the need for content criticality and process services quickly.
- It also offers solutions to solve complicated problems by providing **faster computational speed**, and **enhanced data integrity**.

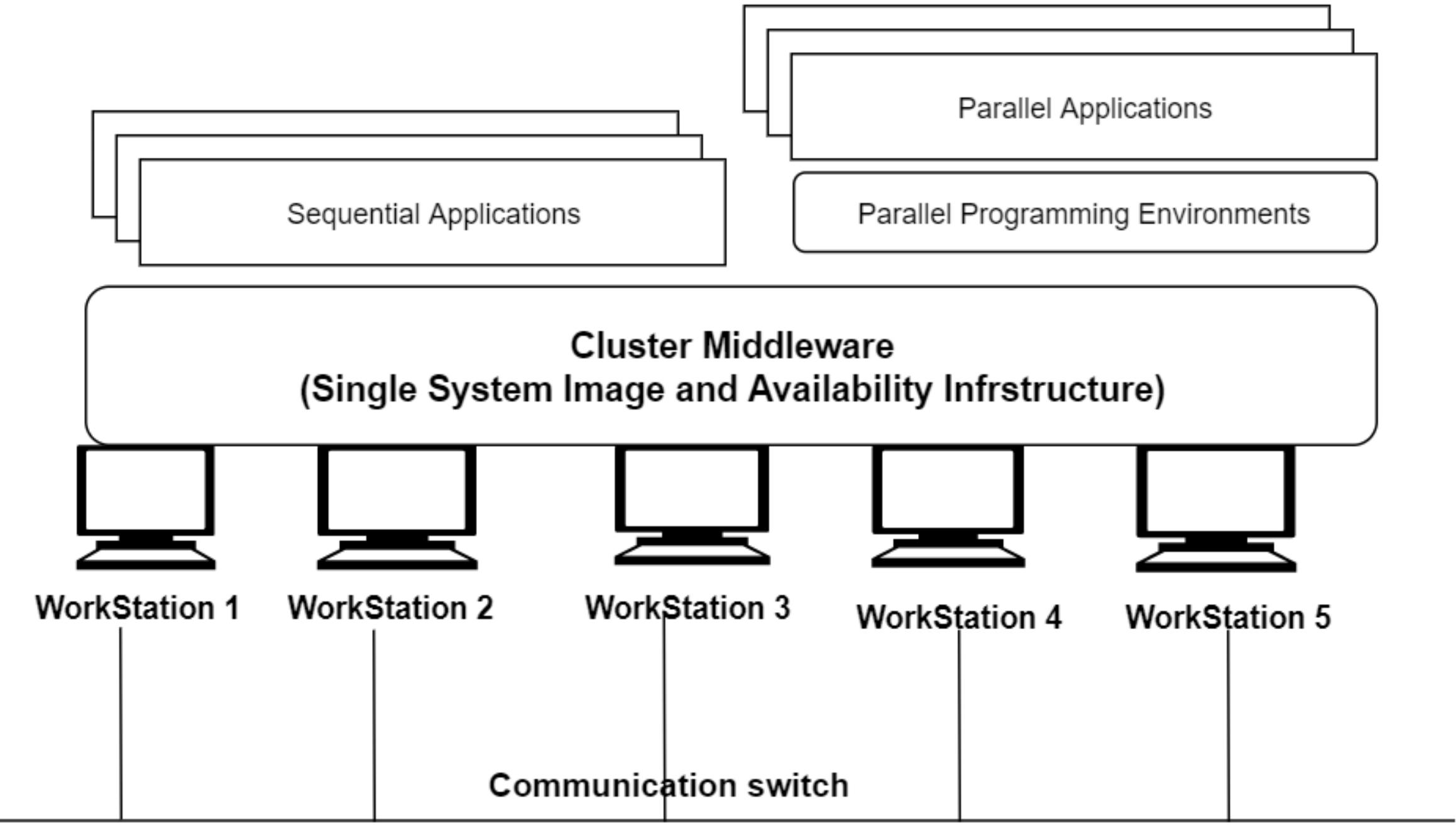
Example applications include:

- Film industry: They require it for rendering extended quality of graphics and cartoons.
- Internet Service Providers look for enhanced availability in a scalable approach, cluster computing will provide this.
- Many of the organizations and IT giants are implementing this technology to augment their scalability, processing speed, availability and resource management at the economic prices.

Cluster Computing (...)

Cluster computing goes with the features of:

- All the connected computers are the same kind of machines.
- They are tightly connected through dedicated network connections
- All the computers share a common home directory.
- Many organizations and IT giants are implementing cluster computing to augment their scalability, processing speed, availability and resource management at the economic prices.



Advantages and Disadvantages of Cluster Computing

Advantages:

- Processing speed
- Extended resource availability
- Expandability
- Flexibility

Disadvantages:

- High Cost (due to the requirement of good hardware and a design)
- Maintenance because more systems are involved.

Utility Computing

- It is a service provisioning model in which a **service provider makes computing resources and infrastructure management available** to the customer as needed, and charges them for specific usage rather than a flat rate.
- Consumers pay providers based on usage (“pay-as-you-go”), similar to the way in which we currently obtain services from traditional public utility services such as **water, electricity, gas, and telephony**.
- It minimizes the associated costs and maximizes the efficient use of resources.
- The advantage of utility computing is that it reduced the IT cost, provides greater flexibility, and easier to manage.

Cloud Computing

Cloud Computing is a general term used to describe a **new class of network based computing** that takes place over the Internet,

- basically a step on from Utility Computing
- a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
- Using the Internet for communication and transport provides hardware, software and networking services to clients

These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).

Cloud Computing (cont....)

In addition, the platform provides on demand services, that are always on, anywhere, anytime and any place.

Pay for use and as needed, elastic

- scale up and down in capacity and functionalities

The hardware and software services are available to

- general public, enterprises, corporations and businesses markets

Note: Cloud computing can be referred to as a method for delivering information technology (IT) services to the users through web-based tools and applications with the help of the internet.

Cloud Computing: Definition

The US National Institute of Standards (NIST) defines cloud computing as follows:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Cloud Computing



- Shared pool of configurable computing resources
- On-demand network access
- Provisioned by the Service Provider

DESIRED FEATURES OF A CLOUD

- 1. Self-Service:** Cloud must allow self-service access so that customers can request, customize, pay, and use services without intervention of human operators.
- 2. Per-Usage Metering and Billing:** Services must be priced on a short-term basis (e.g., by the hour), allowing users to release (and not pay for) resources as soon as they are not needed. Cloud must implement features to allow efficient trading of service such as **pricing, accounting, and billing**. Metering should be done accordingly for different types of service (e.g., storage, processing, and bandwidth) and usage promptly reported, thus providing greater transparency.

DESIRED FEATURES OF A CLOUD (Cont...)

3. Elasticity: To rapidly provide resources in any quantity at any time. Additional resources need to be (scale up and down) :

(a) provisioned, possibly automatically, when an application load increases (**Scale Up**).

(b) released when load decreases (**Scale Down**).

4. Customization: In a cloud scenario, great disparity between user needs is very common. Therefore, resources rented from the cloud must be highly customizable as per the users' requirements.

3-4-5 rule of Cloud Computing

NIST specifies 3-4-5 rule of Cloud Computing

- 3** cloud service models that consists of the particular types of services that you can access on a cloud computing platform.
 - 4** deployment models, which refer to the location and management of the cloud's infrastructure.
 - 5** essential characteristics of cloud computing infrastructure
-

Characteristics of Cloud Computing

5 Essential Characteristics of Cloud Computing

Ref: The NIST Definition of Cloud Computing

<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

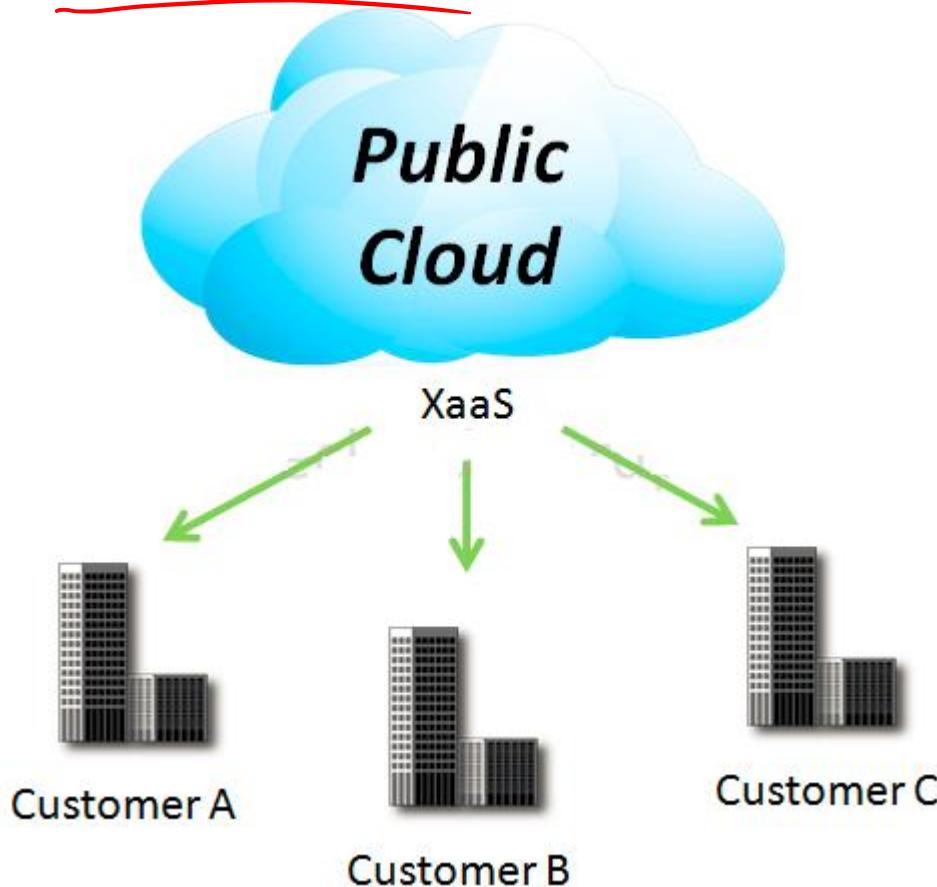


Source: <http://aka.ms/532>

- On demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

4 Deployment Models

1. Public Cloud

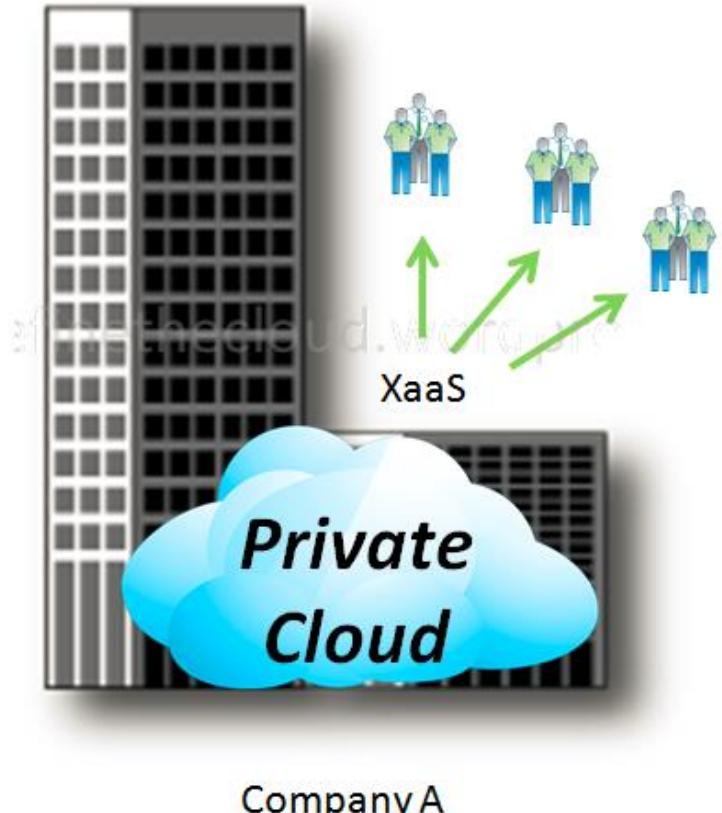


Mega-scale cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

- The physical and IT infrastructure and applications exist at the providers location

4 Deployment Models

2. Private Cloud

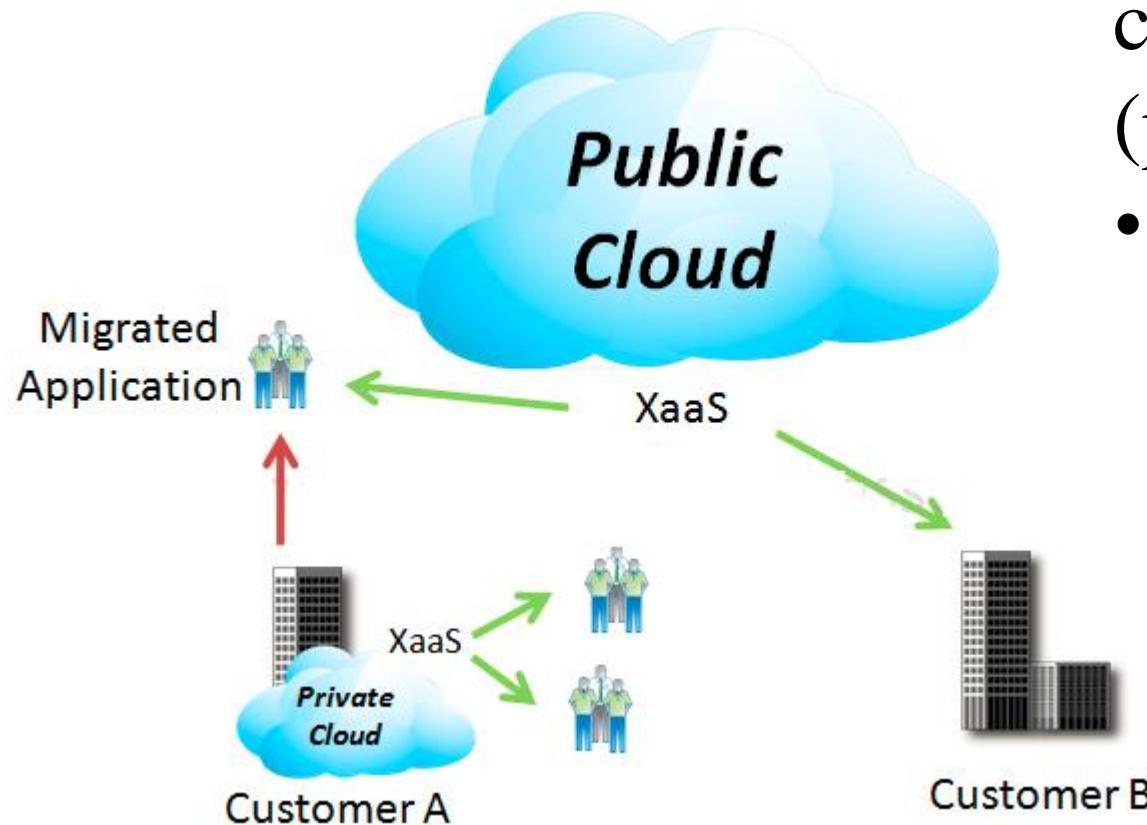


The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist **on premise** or off premise.

- The users of private cloud are the internal business units or divisions.

4 Deployment Models

3. Hybrid Cloud

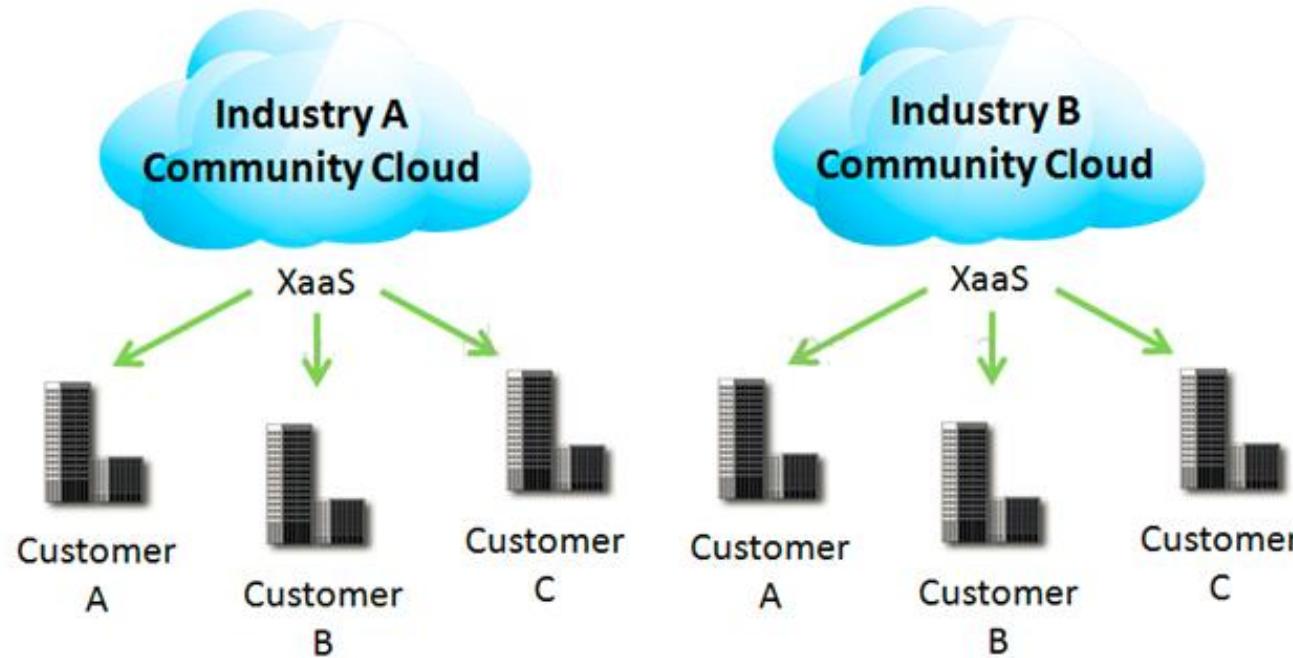


The cloud infrastructure is a composition of two or more clouds (private, community, or public).

- Each cloud retain its features but can share data if required.

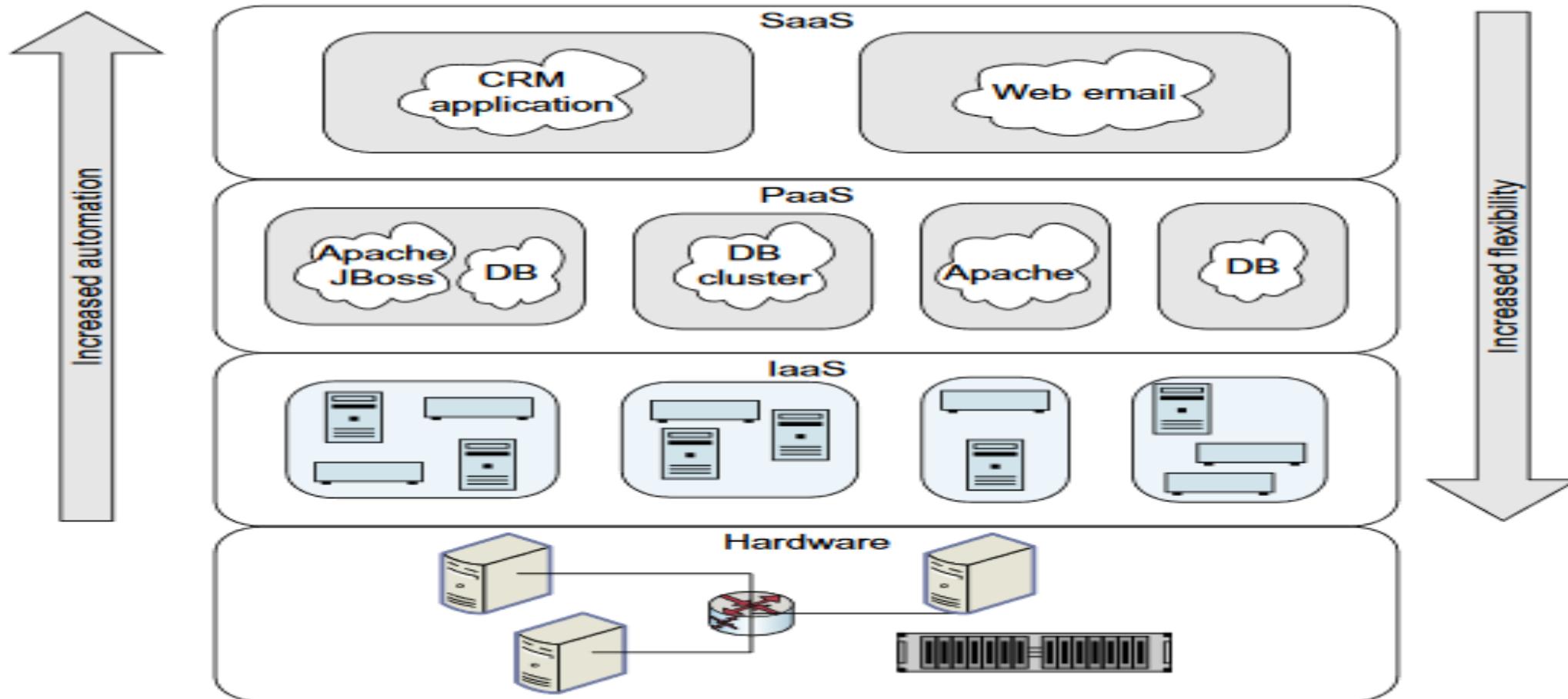
4 Deployment Models

4. Community Cloud



Community Cloud is an infrastructure shared by a specific community of users or organizations **to serve a common function or purpose**. It may be for one organization or for several organizations, but they share common concerns such **as their mission, policies, security, regulatory compliance needs, and so on**. It may be managed by the organizations or a third party and may exist on premise or off premise according to NIST. E.g. OpenCirrus

3 Cloud Service Models



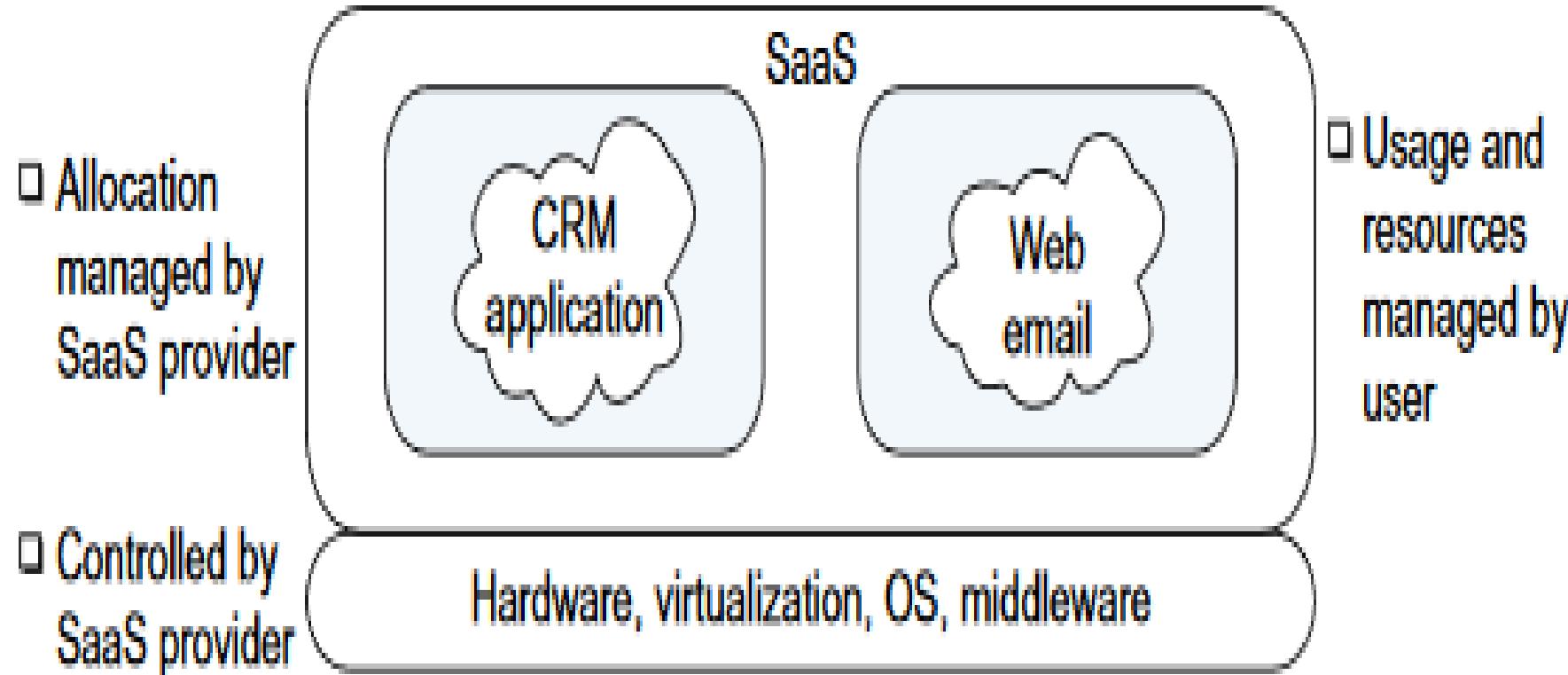
Software as a Service (SaaS)

- In the SaaS model, the **application** is provided to the users **through a thin client interface** (a **browser**, usually), and the customer's responsibility begins and ends with entering and managing its data and user interaction.
- Here, the users don't need to download software and install on their systems. The users do not control the hardware network, security, OS.
- A single instance of the software runs on the cloud and services multiple end users or client organizations.

Examples of SaaS are:

salesforce.com , Google docs, Microsoft Office 365, etc.

Software as a Service (SaaS)



Platform as a Service (PaaS)

It is a service that **can be used to build higher-level services (Applications)**. The user is provided the hardware infrastructure, network, and OS to form a hosting environment. This would be used mostly by developers.

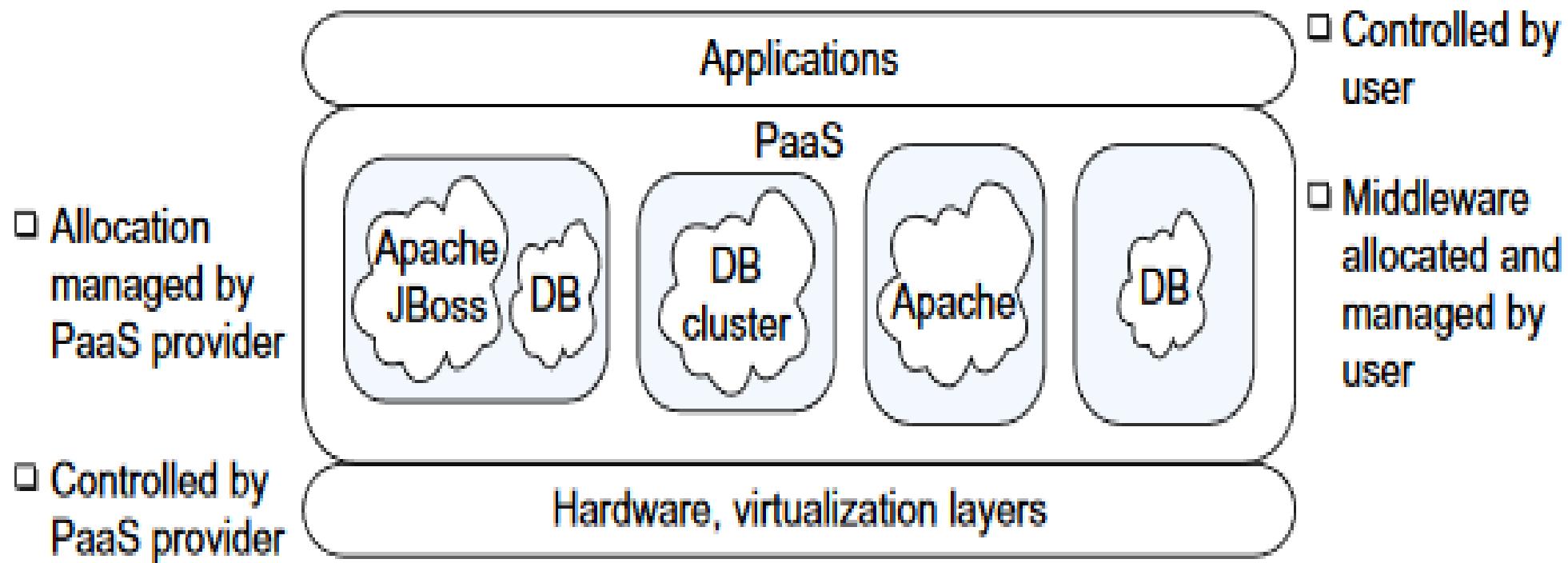
2 Perspectives for PaaS :-

- 1. Producer:-** Someone producing PaaS might produce a platform by integrating an OS, middleware, application software, and even a development environment that is then provided to a customer as a service.
- 2. Consumer:-** Someone using PaaS would see an encapsulated service that is presented to them through an API. The customer interacts with the platform through the API, and the platform does what is necessary to manage and scale itself to provide a given level of service. *Virtual appliances can be classified as instances of PaaS.*

Examples of PaaS are:

AWS Elastic Beanstalk, Google App Engine, and Windows Azure, etc.

Platform as a Service (PaaS)



Infrastructure as a Service (IaaS)

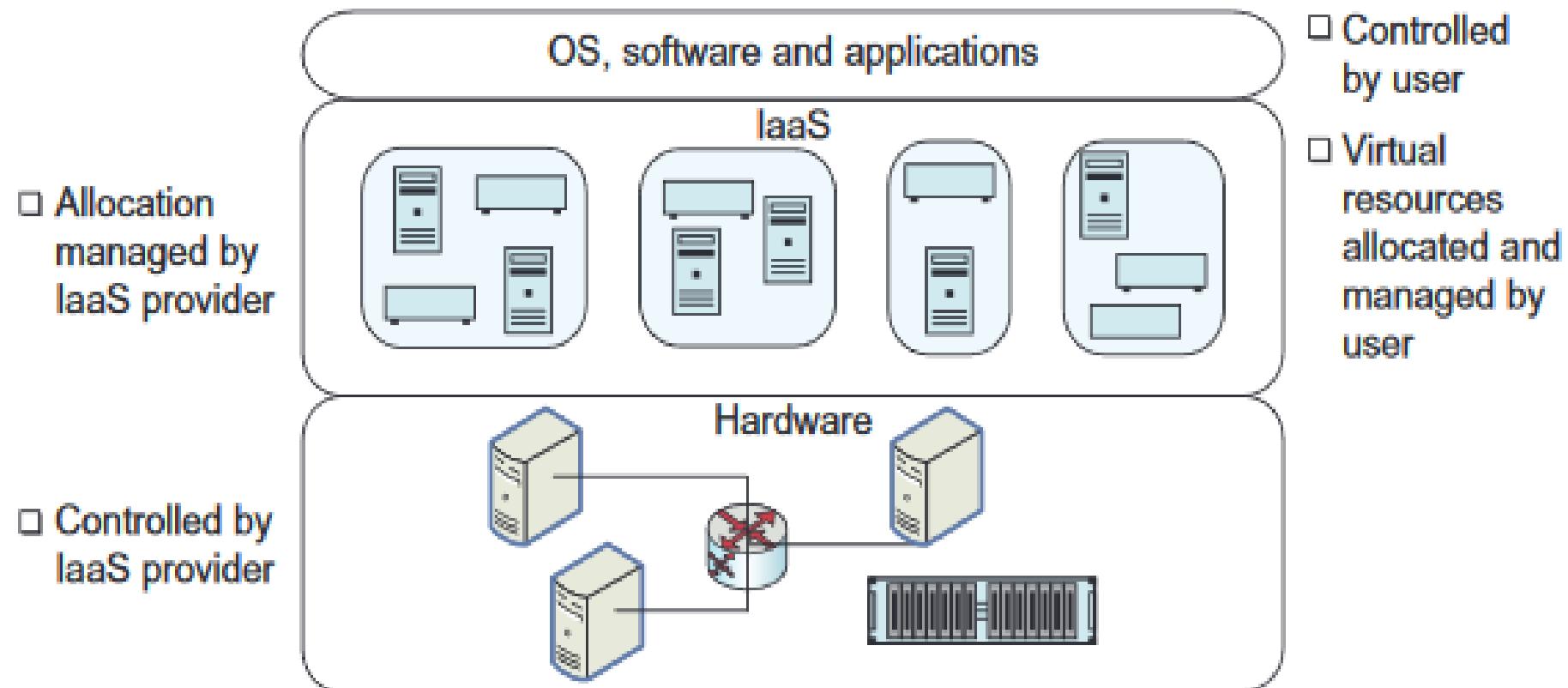
Infrastructure as a service delivers **basic storage and computing capabilities** as standardized services over the network.

- The user gets resources (infrastructure) such as processing power (CPUs), storage, network bandwidth, etc.
- Once the user acquires the infrastructure, he/she controls the OS, data, applications, services, host-based security, etc.

Examples of IaaS are:

Amazon EC2, Google Cloud Platform, Eucalyptus, Rackspace.

Infrastructure as a Service (IaaS)



Cloud Service Provider (CSP)'s Characteristics

- Provide **on-demand provisioning** of computational resources
- Use **virtualization technologies** to lease these resources
- **Provide public and simple remote interfaces** to manage those resources
- Use a **pay-as-you-go cost model**, typically charging by the hour
- Operate data centers large enough to provide a **seemingly unlimited amount of resources** to their clients

The use of “Cloud” Makes refers to:

The “cloud” makes reference to the following two essential concepts:

- 1. Abstraction:** Cloud computing **abstracts the details of system implementation from users and developers.** Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.
- 2. Virtualization:** Cloud computing **virtualizes systems by pooling and sharing resources.** Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.



BITS Pilani



A large, ornate clock tower with multiple levels and arched windows, set against a clear blue sky. The tower is visible in the background of the slide's title area.

Module 2

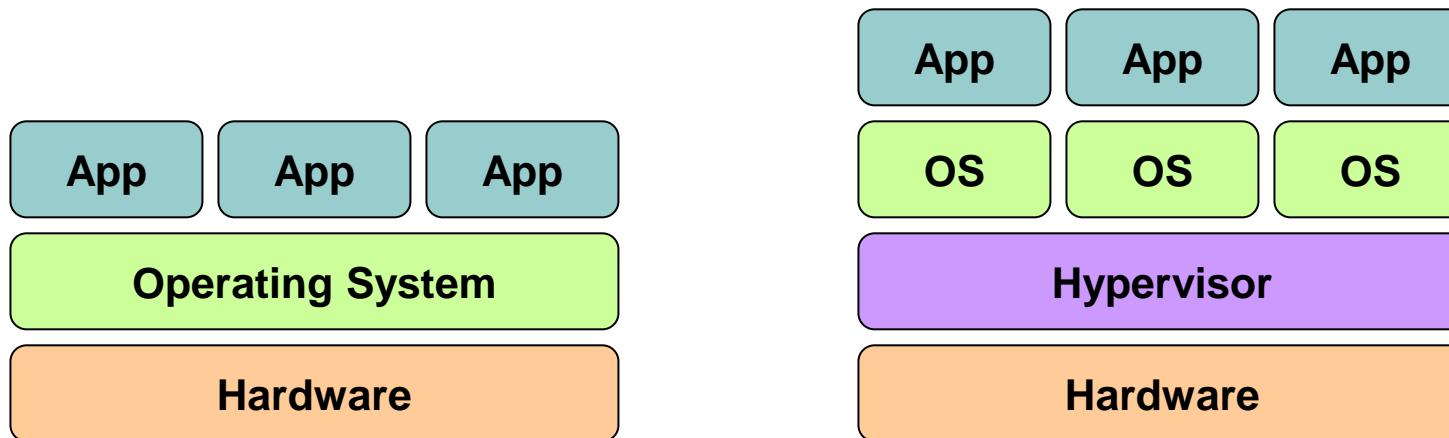
Cloud Computing

CS G527

Dr. D.V.N. Siva Kumar
CSIS Department, BITS Pilani, Hyderabad Campus

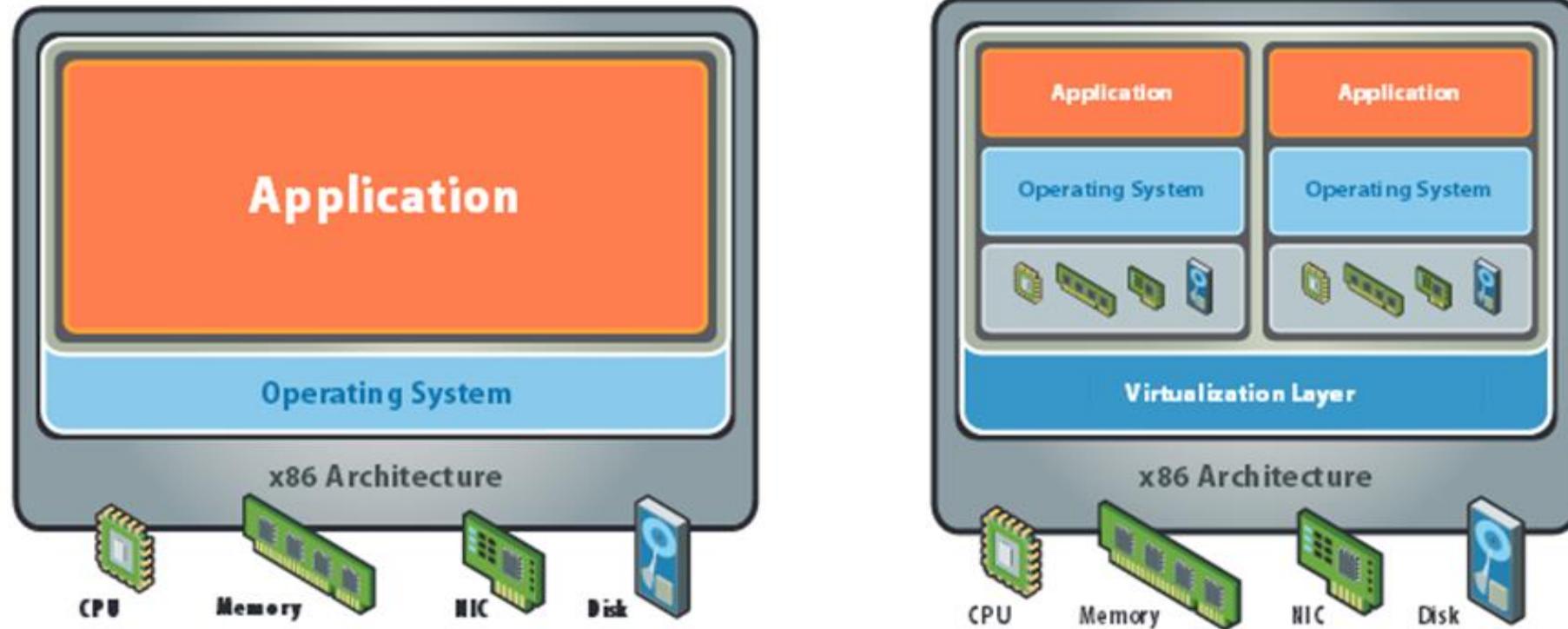
Cloud Infrastructures

Key Technology is Virtualization



Virtualization plays an important role as an enabling technology for datacentre implementation by abstracting compute, network, and storage service platforms from the underlying physical hardware

What is Virtualization?



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.

Key Properties of Virtualization

Partitioning

- Run multiple operating systems on one physical machine.
- Divide system resources between virtual machines.

Isolation

- Provide fault and security isolation at the hardware level.
- Preserve performance with advanced resource controls.

Encapsulation

- Save the entire state of a virtual machine to files.
- Move and copy virtual machines as easily as moving and copying files.

Hardware Independence

- Provision or migrate any virtual machine to any physical server.

Benefits of Virtualization

- More flexible and efficient allocation of resources.
- Improves Security
- Enhance development productivity.
- It lowers the cost of IT infrastructure.
- Remote access and rapid scalability.
- High availability and disaster recovery.
- Pay per use of the IT infrastructure on demand.
- Enables running multiple operating systems.

Goals of Virtualization

1. How to virtualize CPU?.
2. How to virtualize Memory?.
3. How to virtualize I/O?.

Virtualization Providers

- **Microsoft:** Virtual PC, Virtual Server 2005, Hyper-V
- **VMWare:** Vmware Workstation, Vmware Server
- **Oracle:** Oracle VM VirtualBox

Virtual Machine

- Virtual Machine is a result of Virtualization, which typically refers to the creation of virtual machine that can virtualize all of the hardware resources, including **processors, memory, storage, and network connectivity**.
- VM can be treated as **tightly isolated software container with an operating system and application inside**.
- Each self-contained VM is completely independent.
- Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host.”

Hypervisor (Virtual Machine Monitor)

- The software responsible for system virtualization is called the **Virtual machine Monitor (VMM) or Hypervisor**.
- The software is used in two ways (three different structures):
 - **Bare-Metal or Native Hypervisors:** Run directly on the hardware. Examples are VMWare ESX server and KVM
 - **Hosted Hypervisors:** Run on top of existing OS and leverage the features of the underlying OS. Examples are VMWare ESX server,
 - **Hybrid Hypervisors:** Run directly on the hardware, but leverage the features of an existing OS running as a guest. Examples are Xen and Microsoft's Hyper-V

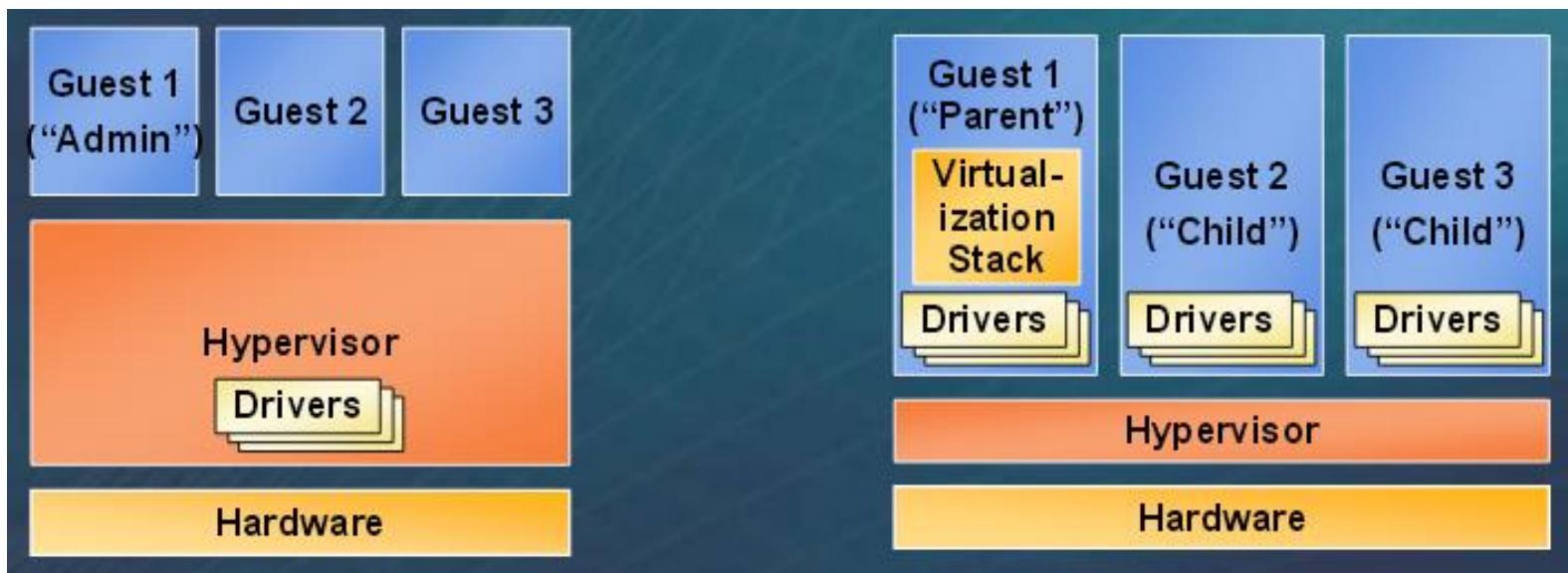
How does Virtualization work?

- Hypervisor separate the physical resources from the virtual environments—the things that need those resources.
- Hypervisors take our physical resources and divide them up so that virtual environments can use them.
- Resources are partitioned as needed from the physical environment to the many virtual environments.
- Users interact with and run computations within the virtual environment (typically called a guest machine or virtual machine).
- When the virtual environment is running and a user or program issues an instruction that requires additional resources from the physical environment, the hypervisor relays the request to the physical system and caches the changes—which all happens at close to native speed (subjected to the type of hypervisor)

Hypervisor

Monolithic versus Microkernelized

- **Monolithic hypervisor**
 - Simpler than a modern kernel, but still complex
 - Contains its own drivers model
- **Microkernelized hypervisor**
 - Simple partitioning functionality
 - Increase reliability and minimize lowest level of the TCB
 - No third-party code
 - Drivers run within guests

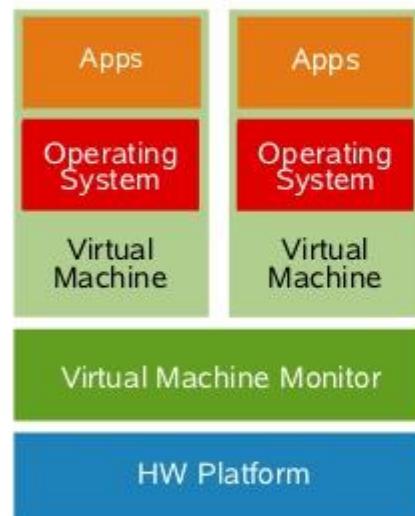


Two Popular Approaches for Server Virtualization

Full & Paravirtualization Overview

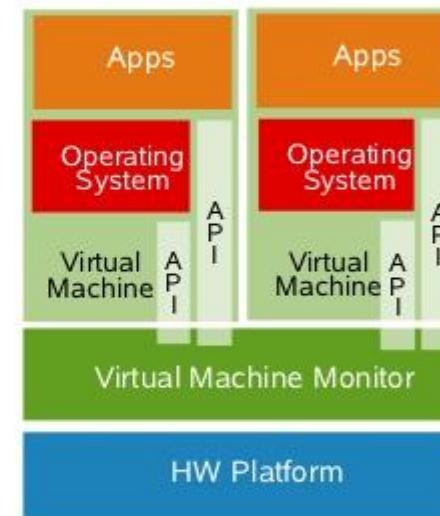


Full Virtualization



Runtime modification of Guest OS:
VMM manages the conflict, then
returns to OS

Paravirtualization



Static modification of Guest OS prior to
runtime: Privileged instruction calls are
exchanged with API functions provided
by the VMM
– Almost no performance degradation
– Significant scalability

Full Virtualization

□ Full virtualization

- In its basic form known as “full virtualization” the hypervisor provides a fully emulated machine in which an operating system can run. **VMWare** is a good example.
- Full virtualization is achieved by using a combination of **binary translation** and **direct execution**. In this type of virtualization, hypervisor translates all privileged instructions (from machine code of guest OS to machine code of host OS) on the fly and caches the results for future use, while user level instructions run unmodified at native speed.
- Guest OS doesn’t see that it is used in an emulated environment.
- The biggest advantage to this approach is its flexibility: one could run a **RISC-based OS** as a guest on an Intel-based host.
- While this is an obvious approach, there are significant performance problems in trying to emulate a complete set of hardware in software.

Advantages and Drawbacks of Full Virtualization

Advantages:

- Isolates VMs from each host OS and from each other.
- Controls individual VM access to system resources, preventing an unstable VM from impacting system performance.
- Total VM portability: VMs have the ability to transparently move between hosts with dissimilar hardware without any problems. A VM running on a Dell Server can be relocated to a HP Server.

Drawbacks:

- Performance due to binary translation of specific privileged instructions.

ParaVirtualization / OS Assisted Virtualization

□ Paravirtualization

- “Paravirtualization,” found in the **XenSource**, open source Xen product.
- **Paravirtualization** uses slightly altered versions of the operating system which allows access to the hardware resources directly as managed by the hypervisor.
- This typically involves replacing any privileged operations that will only run in CPU with calls to the hypervisor (**known as *hypercalls***). The hypervisor in turn performs the task on behalf of the guest kernel and also provides hypercall interfaces for other critical kernel operations such as memory management, interrupt handling, etc.
- Here, the guest OS is modified version and thus run kernel level specific instructions.

ParaVirtualization (Cont...)

- In order to retain flexibility, the guest OS is not tied to its host OS. Drastically different operating systems can be running in a hypervisor at the same time, just as they can under full virtualization.
- Privileged instruction translation by the VMM (Hypervisor) is not necessary.
- Guest OS uses a specialized API to talk to the VMM and execute the privileged instructions.
- In this way, paravirtualization can be thought of as a low-overhead full virtualization.

Advantages and Drawbacks of ParaVirtualization

Advantages:

- Significant performance improvement.

Drawbacks:

- Requires the modified Guest OSes.
- Guest OS could expose the host to security threats due to the direct communication line.

Could you give me an example scenario where one application requires Windows OS?.

Could you give me an example scenario where one application requires Windows OS?

Ans: An application that is developed using .NET requires Windows OS and

What are the few examples of Hypervisors that are available for use.?

Why should a separate computer system is required for hosting an application?

What are the few examples of Hypervisors that are available for use.?

Ans: Oracle Virtual Box and Hyper-V, etc.

Why should a separate computer system is required for hosting an application?

Ans:

- To ensure speed, scalability, and reliability.
- To identify the reasons for the problems encountered while accessing application, etc.

Virtualization Versus Cloud Computing

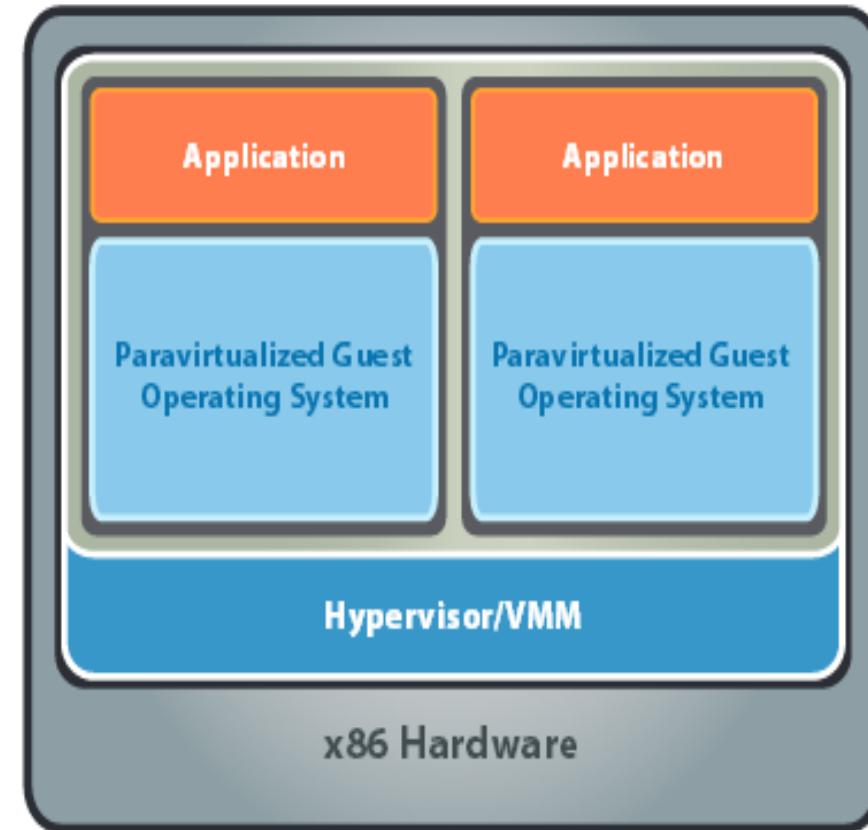
?

Virtualization is software that makes computing environments independent of physical infrastructure, while cloud computing is a service that delivers shared computing resources (software and/or data) on demand via the Internet.

Note: organizations begin by virtualizing their servers and then moving to cloud computing for even greater agility and self-service.

x86 Hardware Virtualization

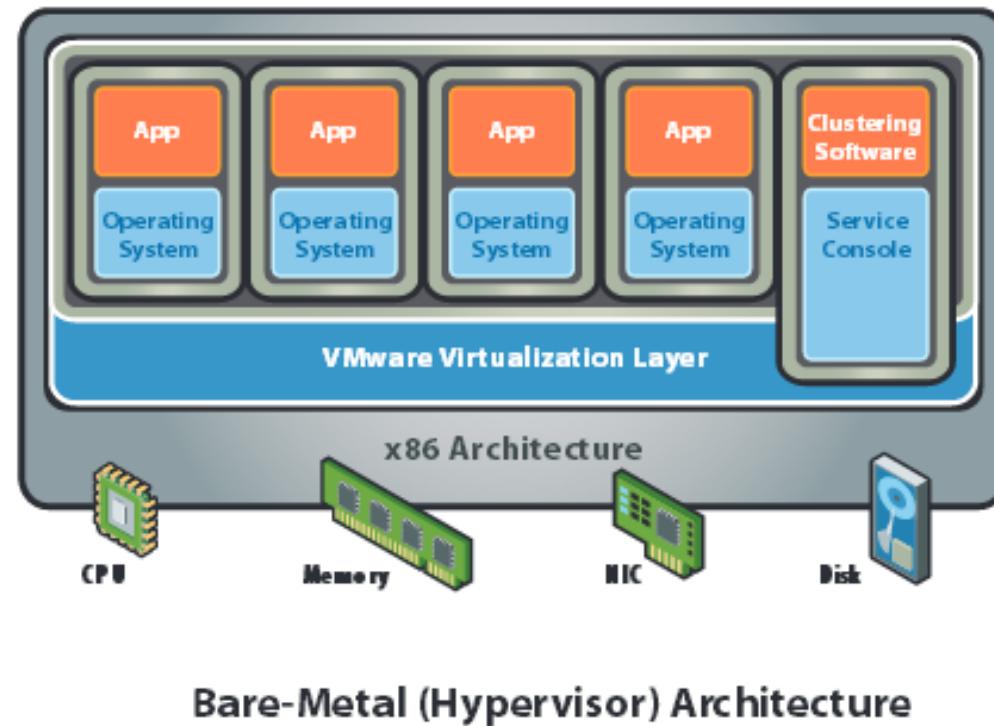
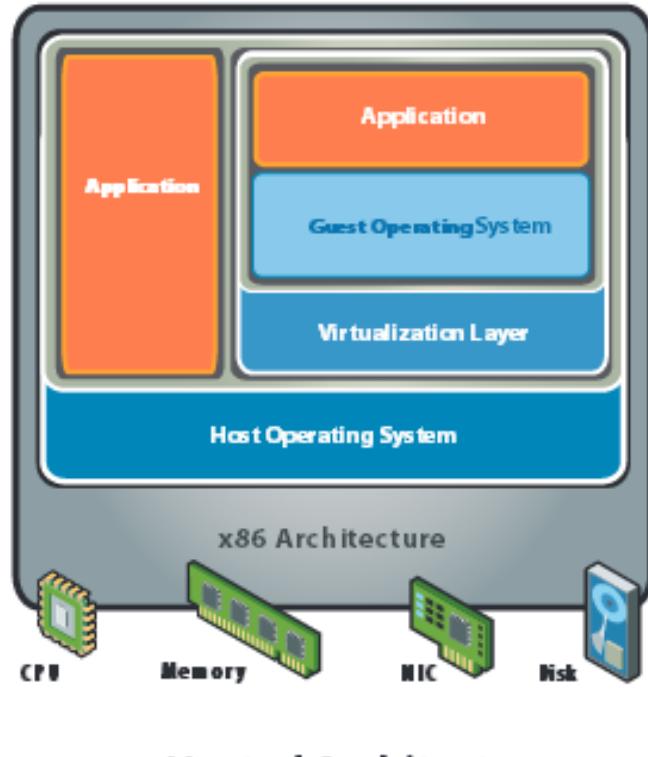
- The latest generation of x86-based systems feature processors with 64-bit extensions supporting very large memory capacities.
- This enhances their ability to host large, memory-intensive applications, as well as allowing many more virtual machines to be hosted by a physical server deployed within a virtual infrastructure.
- The continual decrease in memory costs will further accelerate this trend.



x86 Hardware Virtualization

- For Industry-standard x86 systems, the two approaches typically used with software-based partitioning are
 - Type 1 Hypervisor (also called **bare metal or native**)
 - Type 2 Hypervisor (also known as **hosted hypervisors**)

x86 Hardware Virtualization



Type 1 Hypervisor (also called as Bare-metal hypervisor)

- A [bare-metal hypervisor](#) (Type 1) is a layer of software we install directly on top of a physical server and its underlying hardware.
- There is no software or any operating system in between, hence the name *bare-metal hypervisor*.
- A Type 1 hypervisor is proven in providing excellent performance and stability since it does not run inside Windows or any other operating system.
- Type 1 hypervisors are mainly found in enterprise environments.
- It offers high performance since it has direct access to the underlying hardware (and no other Operating Systems and device drivers to contend with). Due to this, Type 1 Hypervisor is considered to be the best performing and most efficient.
- Two examples of Type 1 Vendors: VMware vSphere, KVM (Kernel-Based Virtual Machine), Citrix Hypervisor (formerly known as Xen Server)

Example Applications of Type 1 Hypervisor

- When High-performance computing is required, where any overhead should be avoided, and hardware components are selected and tuned for maximum performance: e.g., computing clusters for silicon chip design.
- Imagine building a medical imaging device that needs to process huge medical data in real-time and also simultaneously provide an interactive GUI to users. Both the real-time OS and general purpose can be run simultaneously.

Type 2 Hypervisor (also called as Hosted Hypervisors)

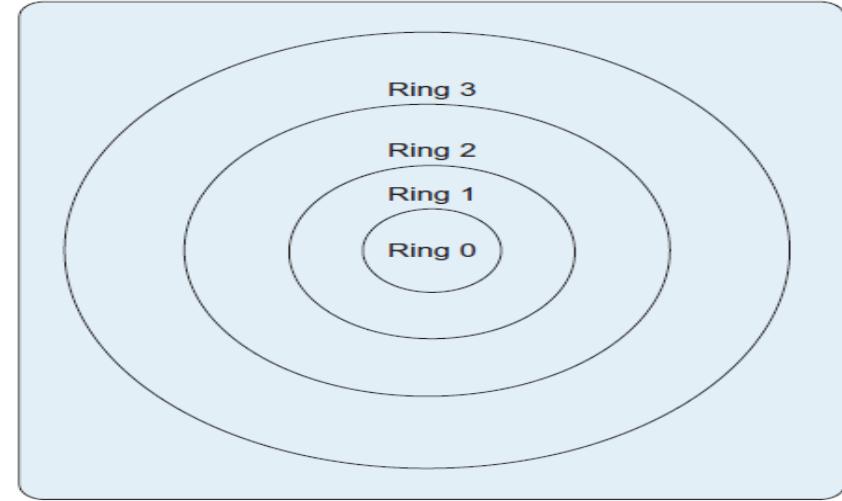
- This type of hypervisor runs inside of an operating system of a physical host machine.
- A **hosted approach** provides partitioning services on top of a standard operating system and supports the broadest range of hardware configurations.
- We call type 2 hypervisors – ***hosted hypervisors***. As opposed to type 1 hypervisors that run directly on the hardware, **hosted hypervisors have one software layer underneath**. In this case we have:
 - A physical machine.
 - An operating system installed on the hardware (Windows, Linux, macOS).
 - A type 2 hypervisor software within that operating system.
 - The actual instances of guest virtual machines.
- Type 2 hypervisors are typically found in environments with a small number of servers.
- Type 2 hypervisors are less efficient than Type 1 since they cannot directly communicate with the hardware due to which they are less efficient than the type 1 hypervisors.

Example Applications of Type 2 Hypervisor

- It is used during the development process. It could be used for testing alpha and beta software as each individual VM is isolated from each other. If one VM (may be a VM running beta software) corrupts the operating system, it will not affect any other VM OS and the host OS.
- This type of hypervisors enables running applications written for several different operating systems on one computer. One VM may be used for running Windows applications and another VM for running Linux applications. In all such cases, type-2 hypervisors are more suitable.

Virtualization Techniques

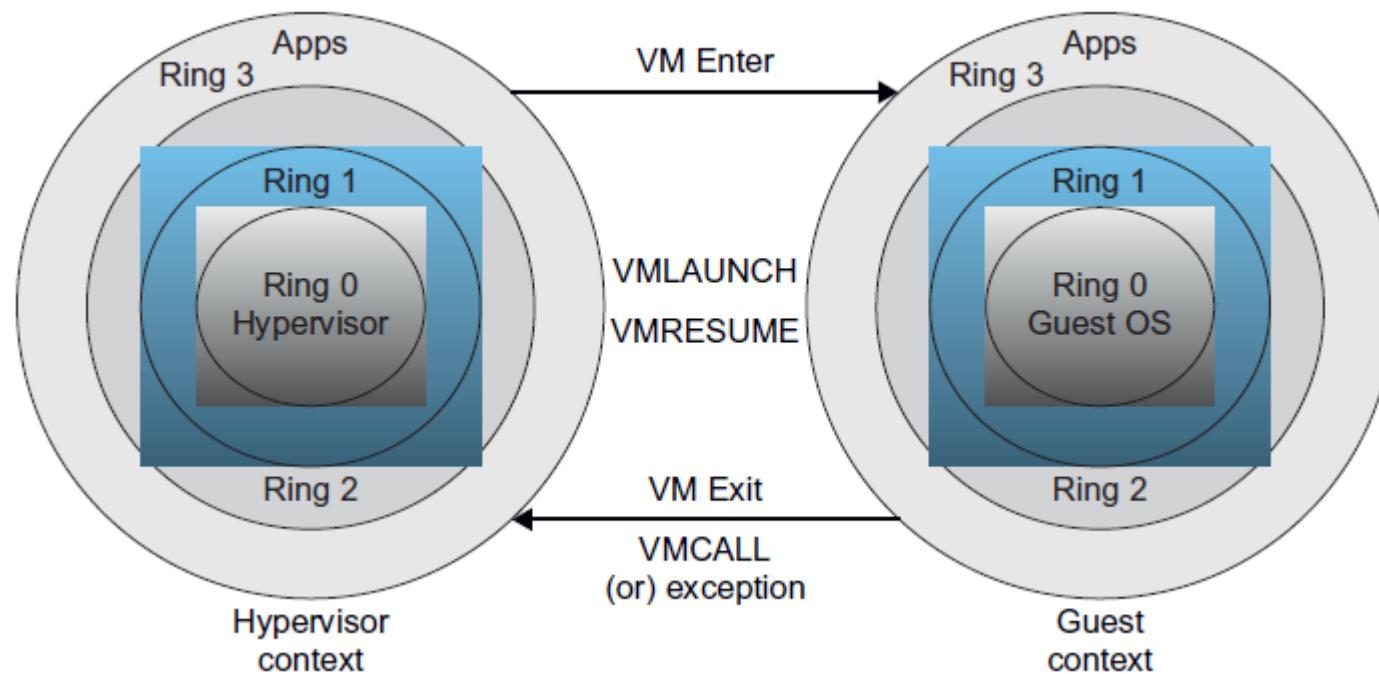
1. Trap and Emulate
2. Binary Translation
3. Paravirtualization
4. Hardware Assisted (Intel and AMD created new processor extensions to support virtualization in the hardware.)



X86 Protection Rings

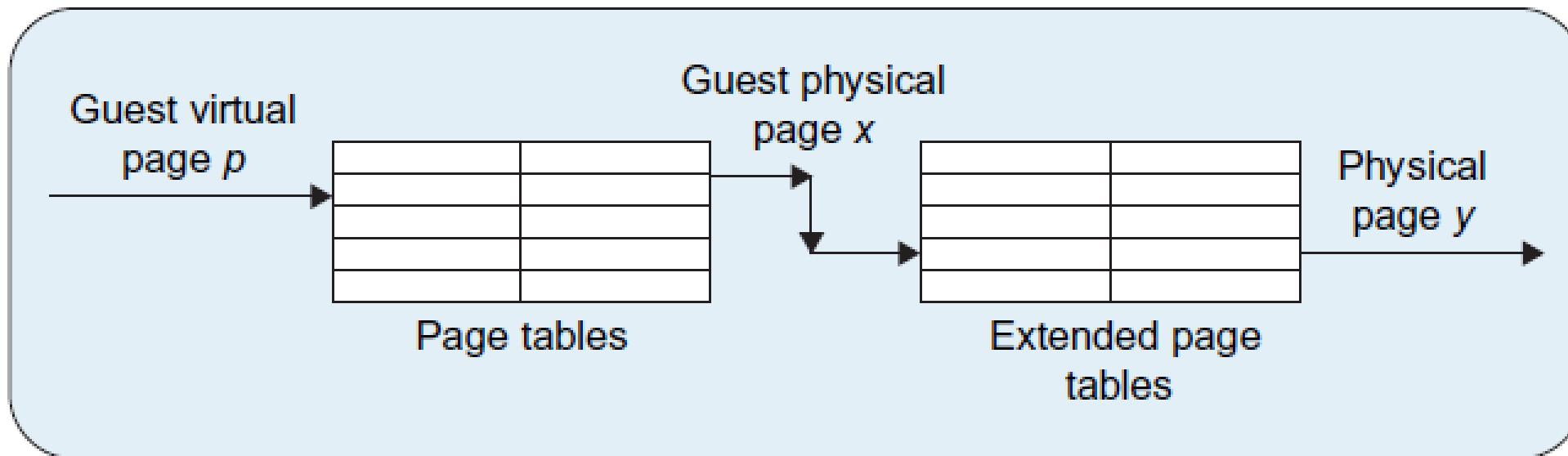
Hardware Support for Processor Virtualization

- VT-x, called VanderPool, represents Intel's technology for virtualization on x86 processors.
- **Two modes** for processor execution: VMX root operation and VMX non-root operation



Hardware Support for Memory Virtualization

Extended page tables.



Hardware Support for IO Virtualization

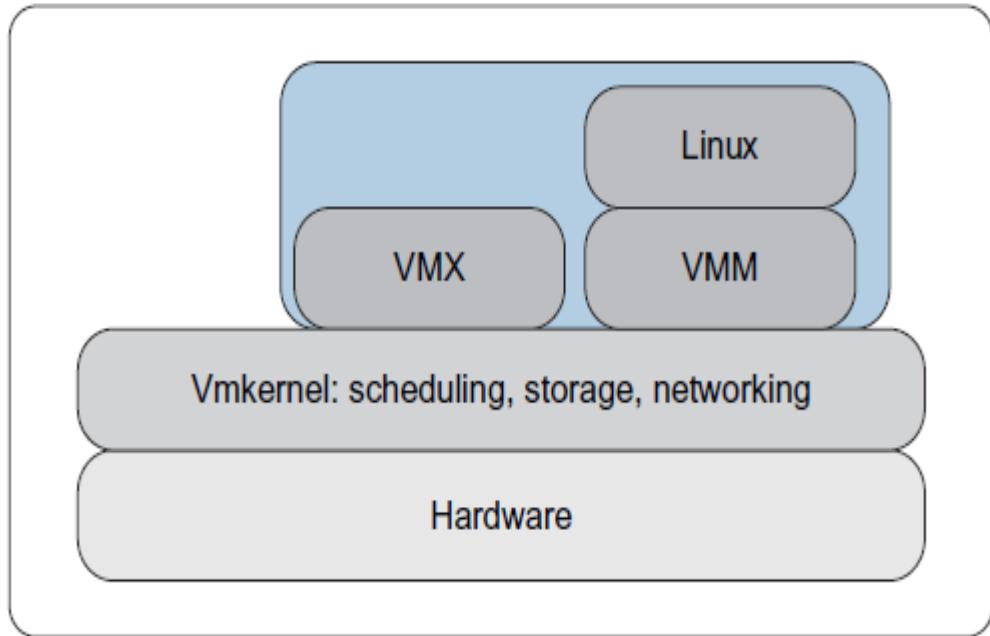
- Intel's VT-d technology can reduce the overhead of I/O virtualization
- It has **two components**, Interrupt Remapping and DMA Remapping
- **DMA remapping** is targeted at eliminating the need for hypervisors to translate guest virtual addresses in I/O commands.
- **Interrupt remapping** is a technology that allows the hypervisor to ensure that interrupts from I/O devices can be delivered directly to the appropriate guest.

Two Popular Hypervisors

1. Vmware
2. Xen

Note: In general, all hypervisors need some operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more—to run VMs.

VMware

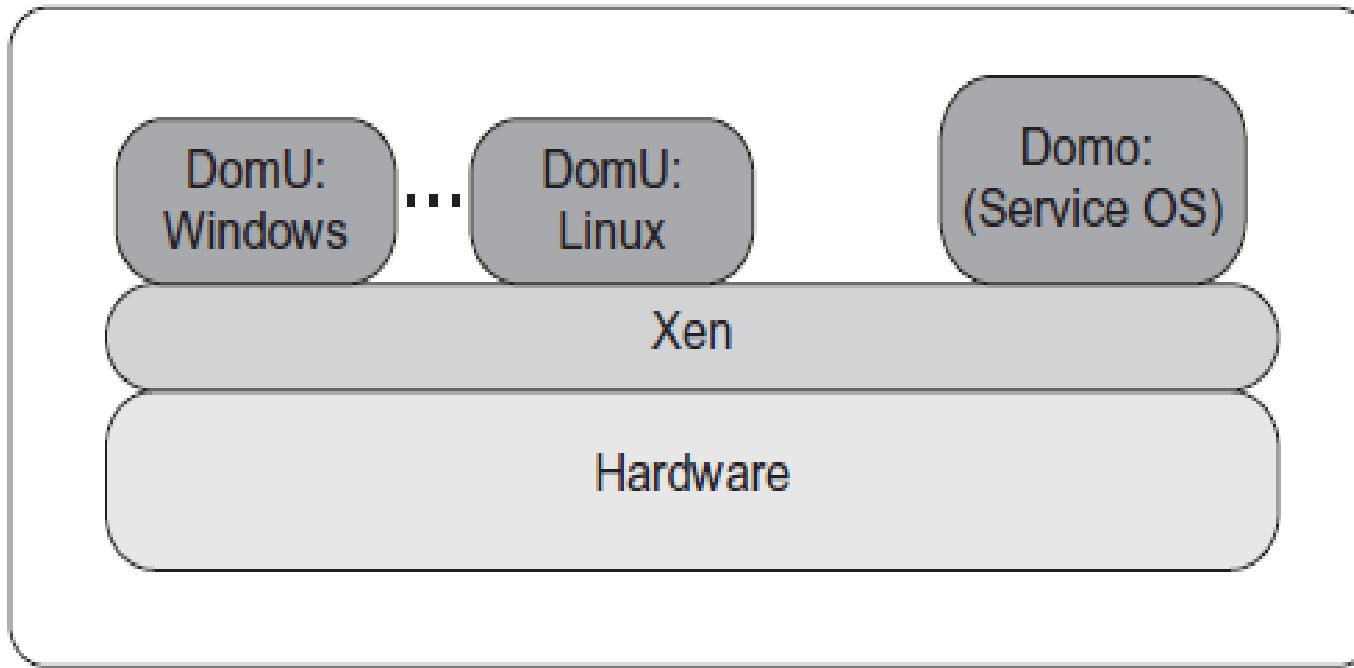


VM virtualization:

- For CPU virtualization, the VMM uses a combination of binary translation and VT-x.
- VMM uses a paravirtualization for I/O virtualization.

High-level architecture of VMware ESX 3i server

Xenserver Architecture

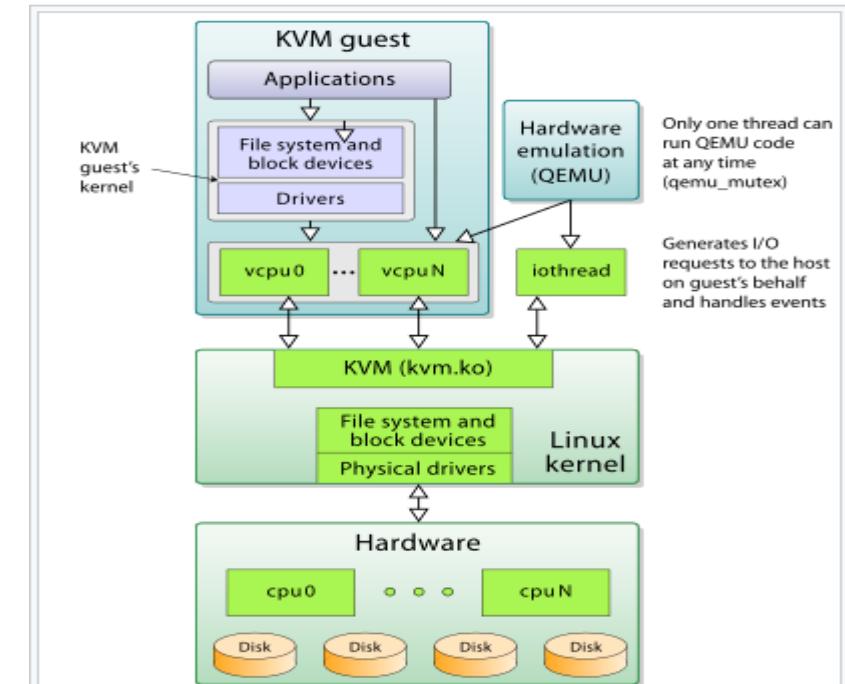


- Guest VMs are called Domains.
- Domain 0 is a specially privileged VM that access to the physical hardware.
- It makes use of paravirtualization for I/O virtualization; I/O requests from any other non-domain 0 VM (called as DomU) are sent to **Dom0**.

KVM (Kernel-based Virtual Machine)

- KVM is an open source virtualization technology built into Linux
- KVM turns Linux into a **hypervisor** that allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).
- KVM provides hardware-assisted virtualization for a wide variety of guest operating systems (Linux, Solaris, Windows, macOS, etc.)

Note: KVM can only be used on a processor with hardware virtualisation extensions such as Intel-VT or AMD-V



Ref: <https://ubuntu.com/blog/kvm-hypervisor>

https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

Advantages of Virtualization

- Security
- Reliability and Availability
- Cost
- Adoptability to workload variations
- Load balancing
- Support for legacy applications

Issues with Virtualization

- Software licensing
- IT training
- Hardware investment
- Interoperability between various vendors



Can we have a single VM on multiple systems?

Can you run multiple virtual machines at once?

Can we have a single VM on multiple systems?

Ans: No, Virtual machines run on a single OS.

Can you run multiple virtual machines at once?

Ans: Yes, you can run multiple virtual machines at once.

Few Questions

Could you give me an example of Hosted Architecture based Virtualization?

Could you give me an example of Bare-Metal (Hypervisor) Architecture based virtualization?

Few Questions

Could you give me an example of Hosted Architecture based virtualization?

Ans: Vmware Workstation and Oracle Virtual Box.

Could you give me an example of Bare-Metal (Hypervisor) Architecture?

Ans: VMware ESX Server employs a hypervisor architecture on certified hardware for data center class performance. Openstack also supports.

Which hypervisor does Google Compute Engine (GCP) use for creating VMs?.

Ans: KVM

Which hypervisor does AWS use for creating VMs?.

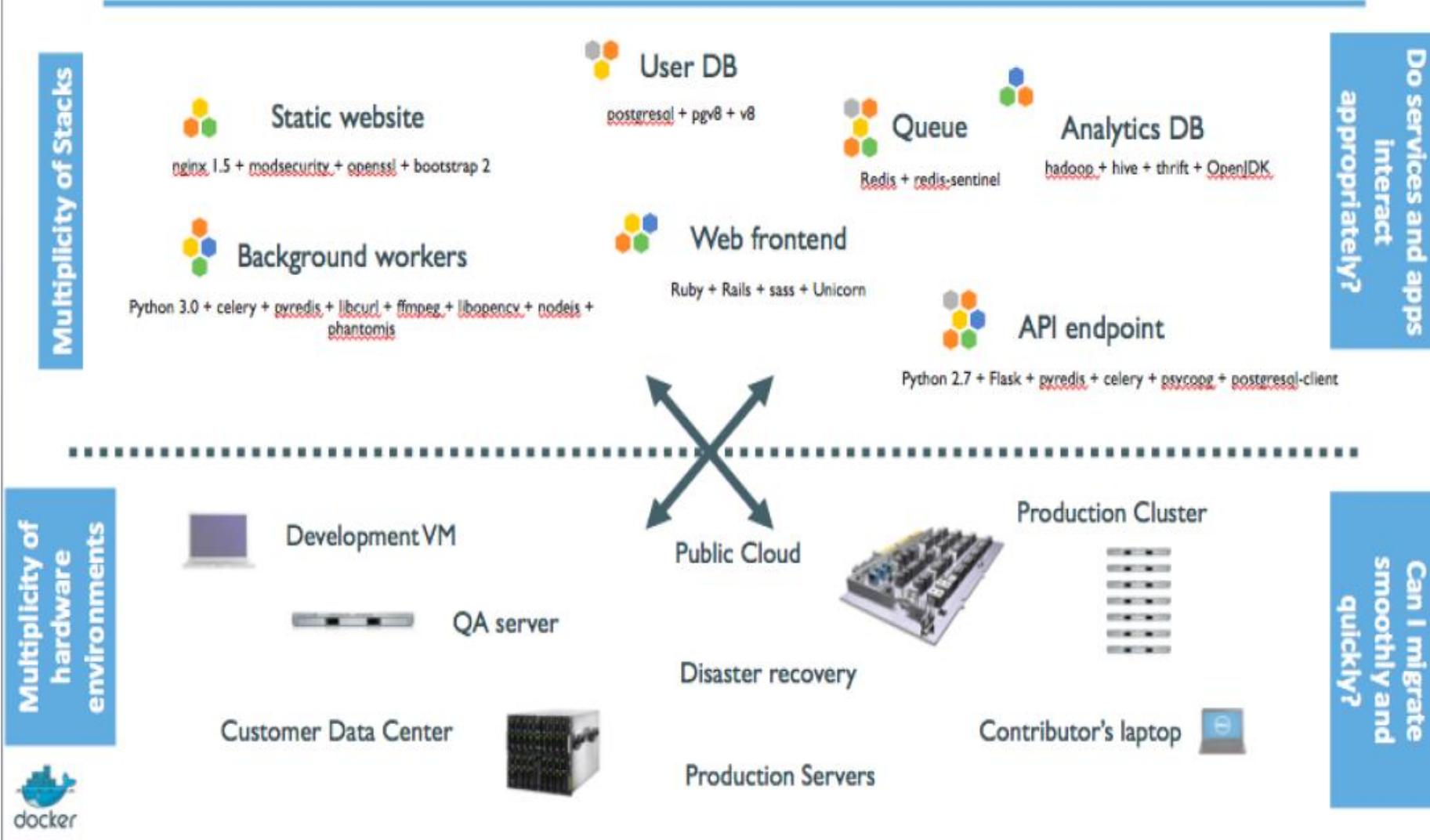
Ans: Modified verisions of KVM (Nitro)

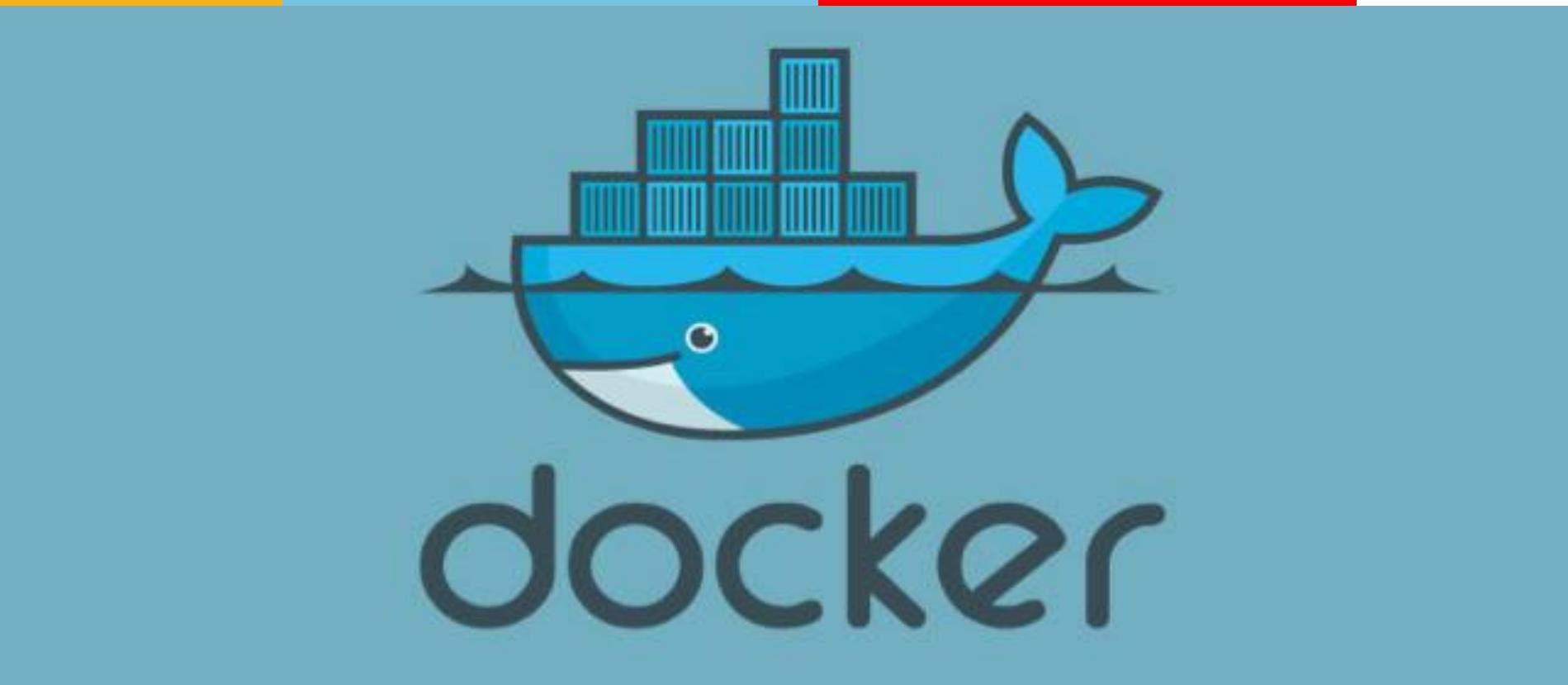
What problems we face with the current virtualization techniques?

Ans:

- Portability of one application from one environment to another across several devices.
- Challenges in Software build up (redeployment or reconfiguration or recompilation of the software whenever we want to ship one package from one environment to other.)
- More booting time for virtual machines to start.
- Possibility of more resource wastage.

Current Problem the Industry is facing





Reference: Getting Started with Docker by Author: Christopher M. Judd (For detailed information about dockers)



Why Dockers are required?

Why Dockers are required? (Cont...)

Docker is a software platform that allows you to build, test, and deploy applications quickly.

It is mainly for:

- Application Level Virtualization
- Build once, deploy anywhere and run anywhere.
- Better collaboration while development of applications.
- A single host can run several applications for proper utilization of resources.

Note1 : Docker enables developers to easily **pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere.**

Note 2: The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

Dockers

- It is an open-source project that automates the deployment of applications inside software containers.
- All applications have their own dependencies, which include both software and hardware resources.
- **Docker is a mechanism that helps in isolating the dependencies per each application by packing them into containers.**
- In terms of technology, it provides portability by running the same applications in different environments.
- Containers are scalable and safer to use and deploy as compared to regular approaches.

Note: The Dockers can also be called as container-based virtualization or Lightweight virtualization.

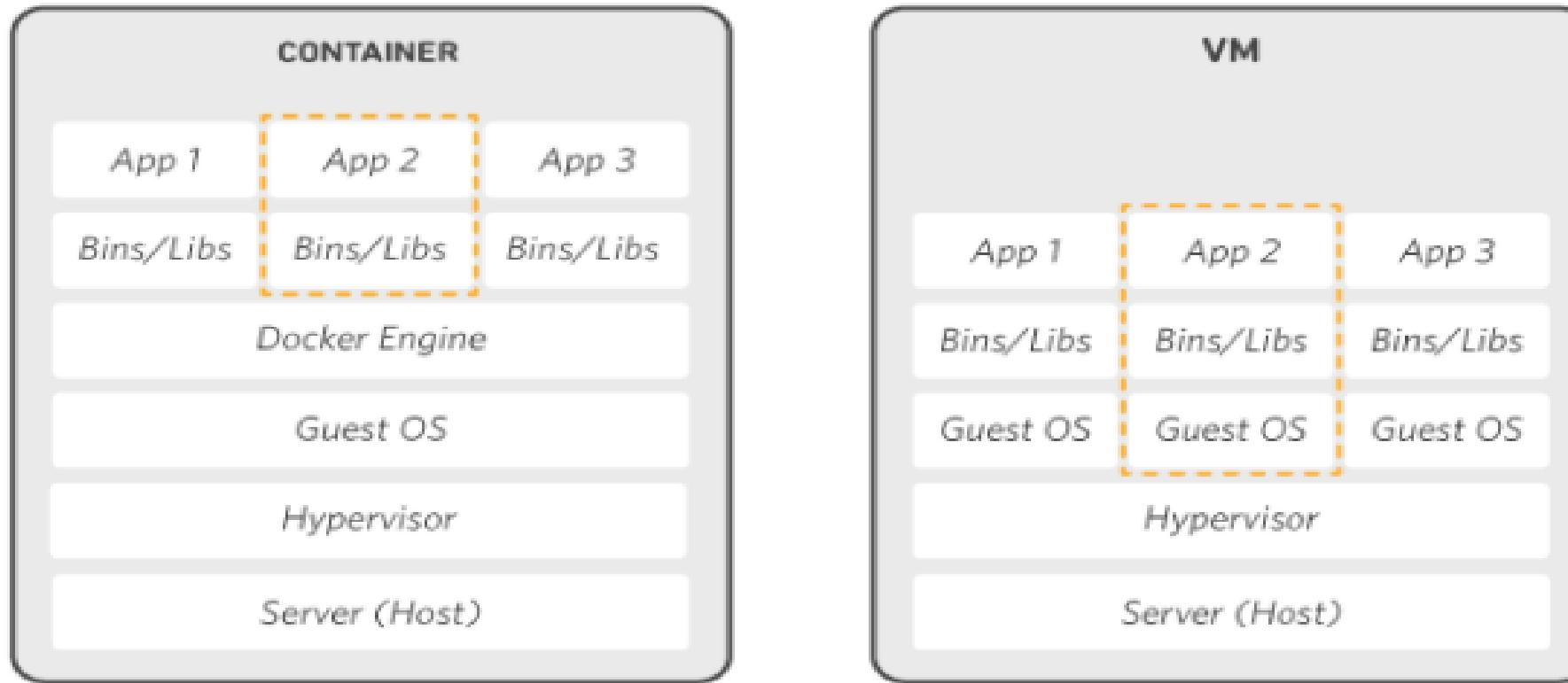


How are Docker Containers different from a Virtual Machine?

- Virtual machines have a full OS with its own memory management installed with the associated overhead of virtual device drivers.
- Docker containers are executed with the Docker engine rather than the hypervisor.
- Containers are therefore smaller than Virtual Machines and enable faster start up with better performance, less isolation and greater compatibility possible due to sharing of the host's kernel.



How are Docker Containers different from a Virtual Machine?



Ref: <https://aws.amazon.com/docker/>

Docker Architecture

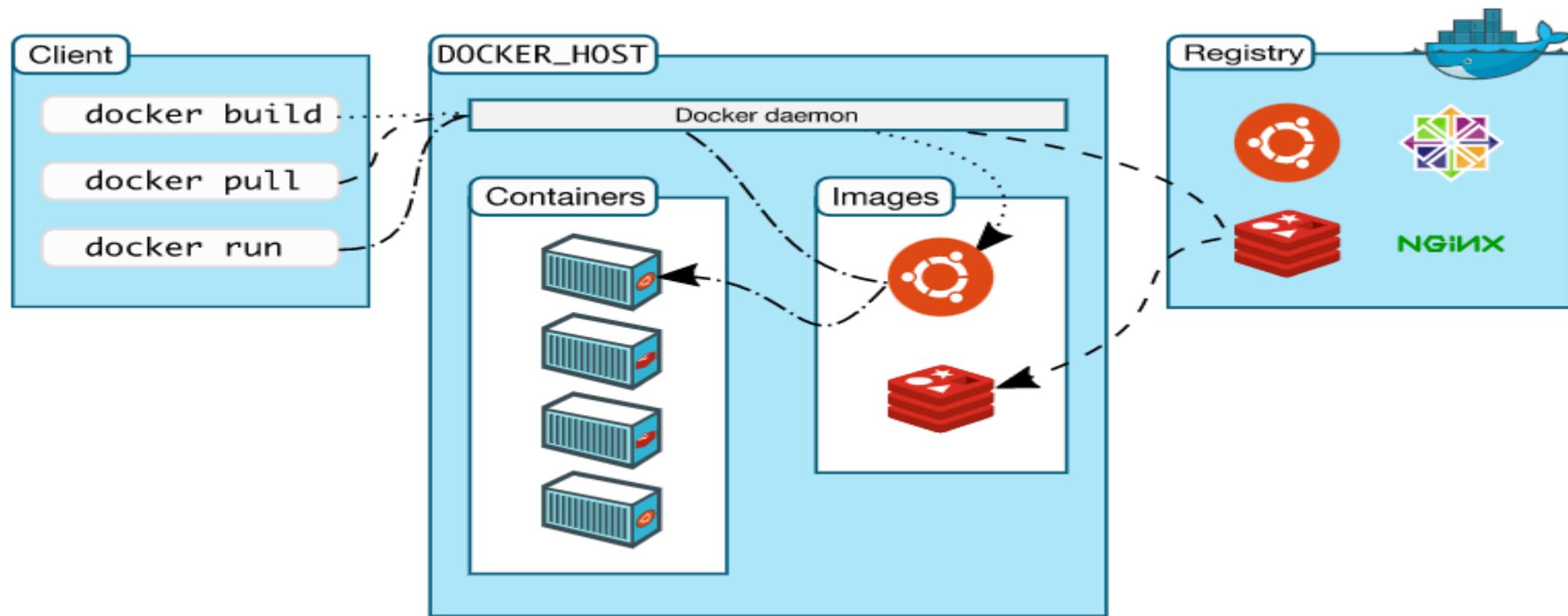


Image and Container

Image: An *image* is a read-only template with instructions for creating a Docker container. For example, build an image which is based on the ubuntu image, but install the Apache web server and our application, as well as the configuration details needed to make our application run.

Note: You might create your own images or you might only use those created by others and published in a registry.

Container:

- Docker provides the ability to package and run an application in a loosely isolated environment called a container.
- A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI.
- We can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

Docker Components

Docker for Mac – To run Docker containers on the Mac OS.

Docker for Linux – To run Docker containers on the Linux OS.

Docker for Windows – To run Docker containers on the Windows OS.

Docker Engine – For building Docker images and creating Docker containers.

Docker Hub – It is the registry which is used to host various Docker images.

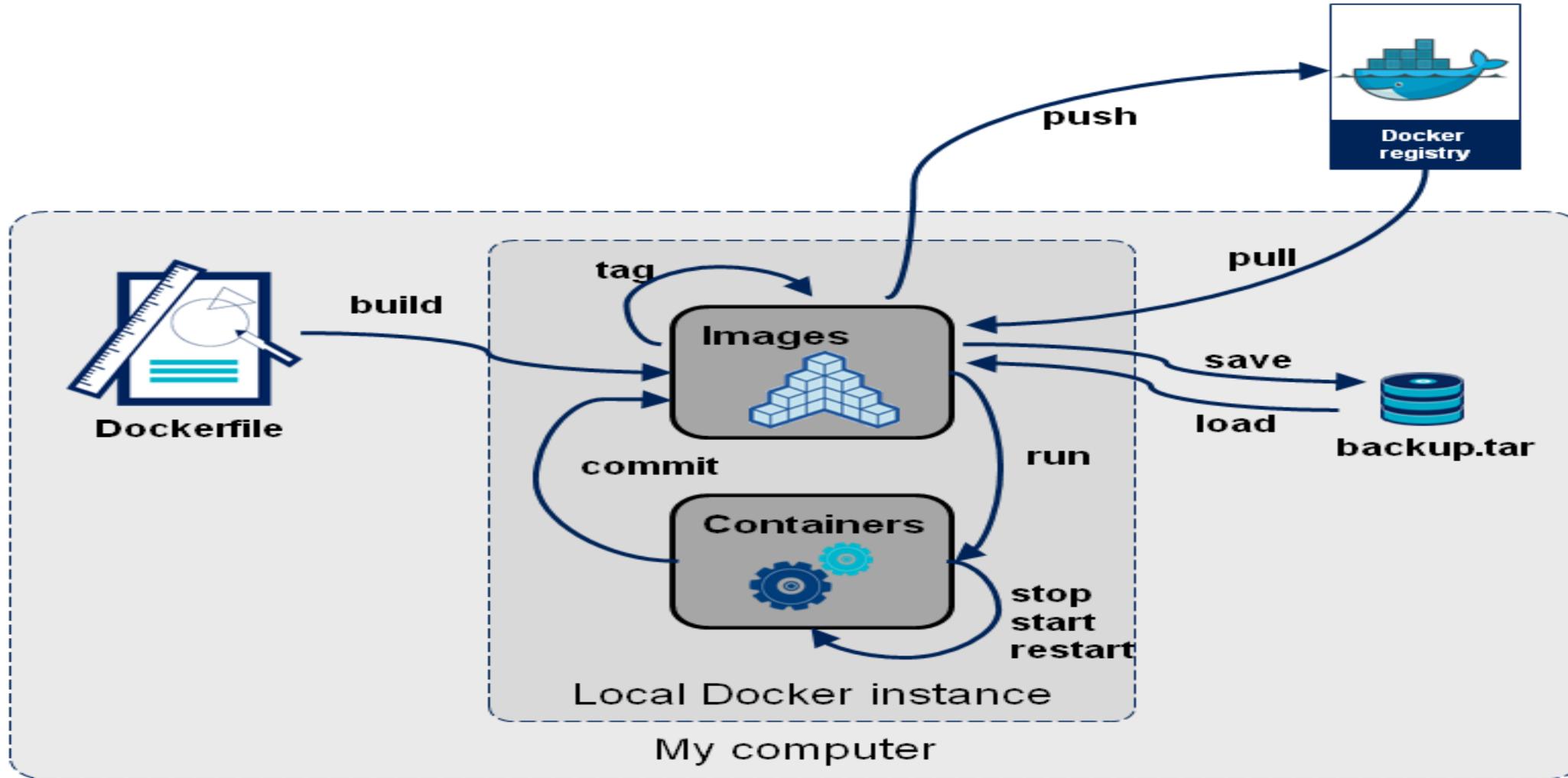
Docker Compose – This is used to define applications using multiple Docker containers.

Docker Features

- It has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- Can be deployed anywhere, on any physical machine (Mobile, System, Server), virtual machine and even in cloud.
- Higher scalability since containers are lightweight.



Docker Container Lifecycle



Dockerfile

- Like a Makefile (shell script with keywords)
- Extends from a Base Image
- Results in a new Docker Image
- Imperative, not Declarative
- A Docker file lists the steps needed to build an images
- docker build is used to run a Docker file
- Can define default command for docker run, ports to expose, etc



Example of a Dockerfile

file | 15 lines (11 sloc) | 0.475 kb

Open Edit Raw Blame History Delete

```
1 FROM ubuntu:12.04
2
3 RUN apt-get update
4
5 # Make it easy to install PPA sources
6 RUN apt-get install -y python-software-properties
7
8 # Install Oracle's Java (Recommended for Hadoop)
9 # Auto-accept the license
10 RUN add-apt-repository -y ppa:webupd8team/java
11 RUN apt-get update
12 RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
13 RUN apt-get -y install oracle-java7-installer
14 ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
```



Differences between Virtual Machines and Docker Containers

	Virtual Machines	Docker Containers
Isolation Process Level	Hardware	Operating System
Operating System	Separated	Shared
Boot up time	Long	Short
Resources usage	More	Less
Pre-built Images	Hard to find and manage	Already available for home server
Customised preconfigured images	Hard to build	Easy to build
Size	Bigger because they contain whole OS underneath	Smaller with only docker engines over the host OS
Mobility	Easy to move to a new host OS	Destroyed and recreated instead of moving.
Creation time	Longer	Within seconds

Security concerns with Dockers

- Kernel exploitations
- Denial-of-service attacks
- Container breakouts
- Poisoned images

Reference: <https://www.oreilly.com/content/five-security-concerns-when-using-docker/>

When to use Virtualization?

When to use Containerization?

Video Lectures of Demonstrating Hypervisors

- <https://www.youtube.com/embed/phCWC7AgxYM?modestbranding=1&showsearch=0&autohide=1&showinfo=0&rel=0&frameborder=0>
(Demonstration of installing Ubuntu OS using hypervisor called “Oracle Virtual Box”)
- <https://www.youtube.com/embed/EYqwMulUKWY?modestbranding=1&showsearch=0&autohide=1&showinfo=0&rel=0&frameborder=0>
(Demonstration of installing Ubuntu OS using another hypervisor called “Microsoft Hyper-V”)
- https://www.princeton.edu/~rblee/ELE572Papers/Fall04Readings/secureOS/popek_virtualizable.pdf (Formal Requirements for Virtualizable Third Generation Architectures)

References

- <https://blackberry.qnx.com/content/dam/qnx/whitepapers/2017/what-is-a-hypervisor-and-how-does-it-work-pt1.pdf>
- <https://www.citefactor.org/journal/pdf/A-Virtualization-Approach-in-Smart-phone-Using-Cloud-computing-for-machine-to-machine-Communication.pdf>
- <http://dsc.soic.indiana.edu/publications/virtualization.pdf>
- <https://www.vmware.com/in/solutions/virtualization.html>
- <https://courses.cs.washington.edu/courses/cse451/18sp/readings/virtualization.pdf>
- **Getting Started with Docker by Author: Christopher M. Judd (For detailed information about Docker architecture)**
- <https://reader.elsevier.com/reader/sd/pii/S0140366417300956?token=1BD05F31679AFAE6D696EDCF93DA910B51D2CDB3CDEA5A1B88AB87B6E50DD62AA0DB365B7AB9CD83A29F0C9568058F34&originRegion=eu-west-1&originCreation=20220205092527> (Dockers)
- <https://reader.elsevier.com/reader/sd/pii/S1877050920311315?token=12D4662BF758D895621AEC6D7BFA5017249B01073B19B65D815E5DA9BF802399AEC3308F55520B7B4B19FFD993C62B46&originRegion=eu-west-1&originCreation=20220205092613> (Dockers)

Cloud Computing

CSI ZG527 / SS ZG527 / SE ZG527



BITS Pilani

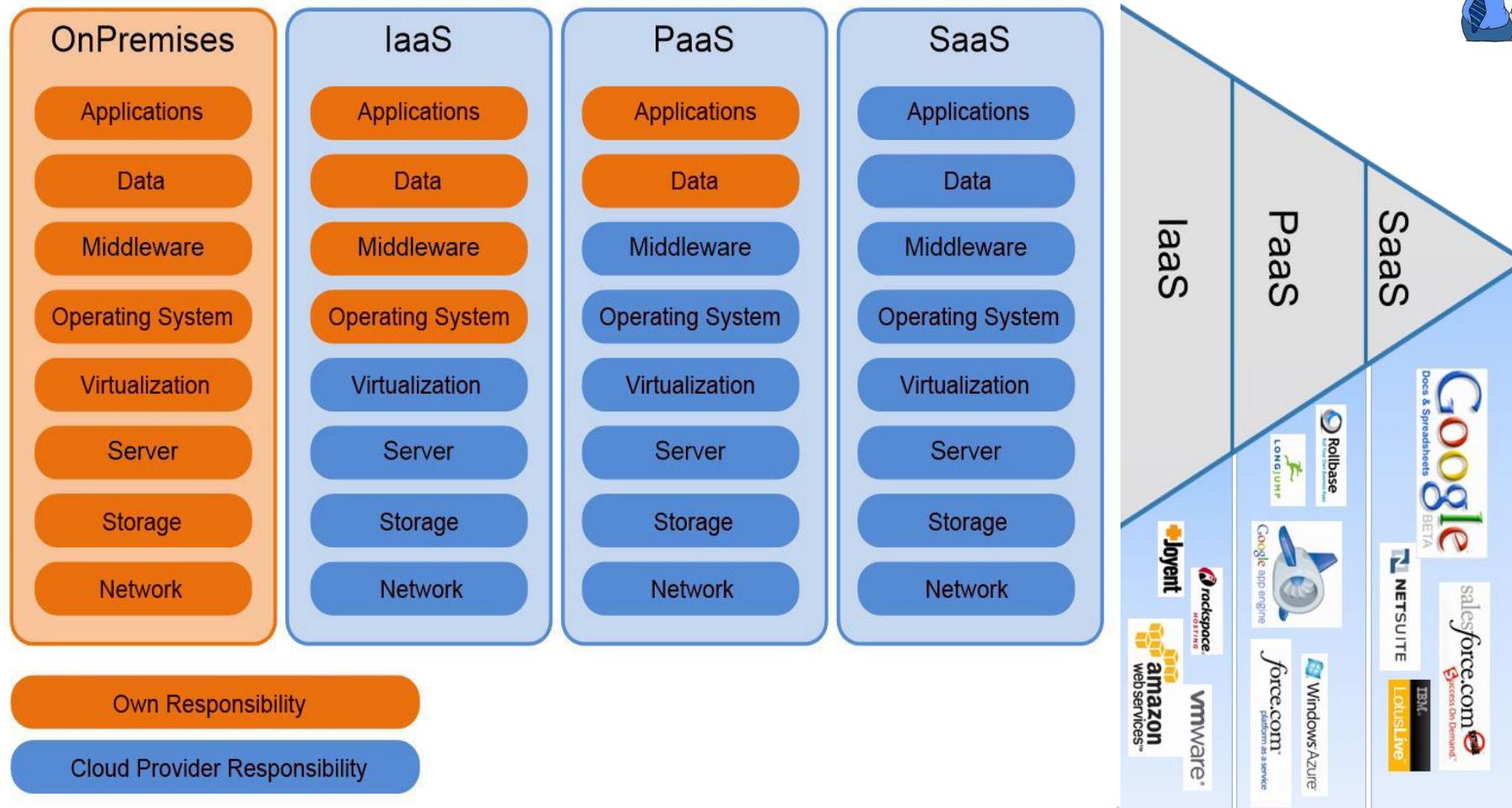
Module 3 Infrastructure as a Service



heard of 3 models of Cloud Computing?



Yes, Yes, IaaS, PaaS and SaaS



Infrastructure as a Service (IaaS)

- Under the IaaS cloud computing model, **cloud service providers make computing and storage resources (such as servers and storage) available as a service.**
- This offers maximum flexibility for users to work with the cloud infrastructure, wherein exactly how the virtual computing and storage resources are used is left to the cloud user.

Features of IaaS

- Geographic Presence
- User Interfaces and Access to Servers
- Advance Reservation of Capacity
- Automatic Scaling and Load Balancing
- Automatic Scaling and Load Balancing
- Service-Level Agreement (SLA)
- Hypervisor and Operating System Choice

The value of IaaS

For businesses, the greatest value of IaaS is through a concept known as ***cloudbursting***—the process of off-loading tasks to the cloud during times when the most compute resources are needed.

To take advantage of IaaS in this capacity, IT departments must be able to build and implement the software that handles the ability to re-allocate processes to an IaaS cloud.

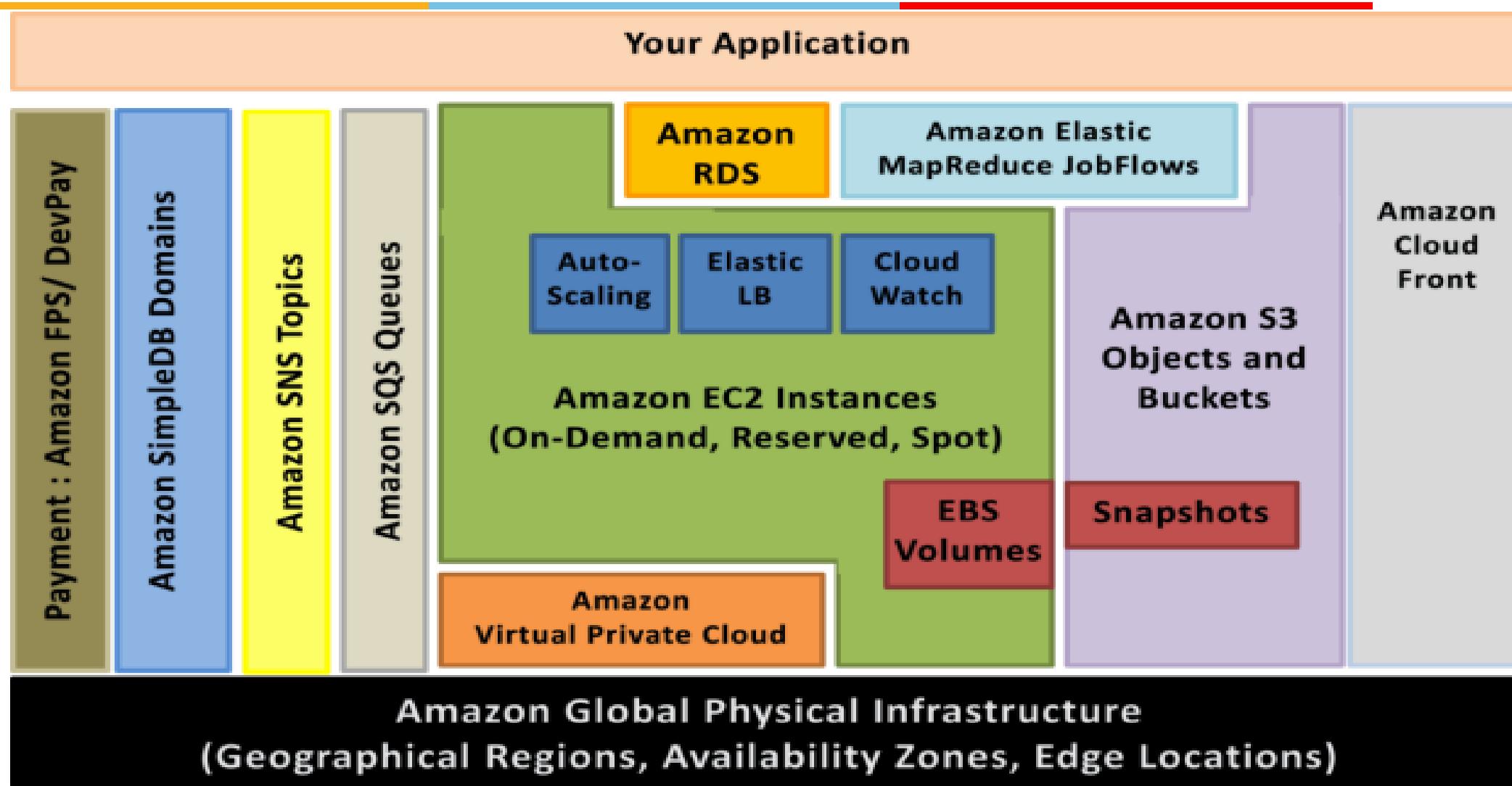
There are **four important considerations** to build and implement software that can manage such reallocation processes.

4 considerations:

- Developing for a specific vendor's proprietary IaaS could prove to be a costly mistake.
 - The complexity of well-written resource allocation software is significant and do not come cheap
 - What will you be sending off to be processed in the cloud? Sending data such as personal identities, financial information, and health care data put an organization's compliance at risk
 - Understand the dangers of shipping off processes that are critical to the day-to-day operation of the business.
-
- <http://www.ibm.com/developerworks/cloud/library/cl-cloudservices1iaas/>

Feature Comparison of Virtual Infrastructure Managers												
	License	Installation Platform of Controller	Client UI, API, Language Bindings		Backend Hypervisor(s)	Storage Virtualization	Interface to Public Cloud	Virtual Networks	Dynamic Resource Allocation	Advance Reservation of Capacity	High Availability	Data Protection
			Platform	Language								
Apache VCL	Apache v2	Multi-platform (Apache/PHP)	Portal, XML-RPC	VMware ESX, ESXi, Server	No	No	Yes	No	Yes	No	No	No
AppLogic	Proprietary	Linux	GUI, CLI	Xen	Global Volume Store (GVS)	No	Yes	Yes	No	Yes	Yes	Yes
Citrix Essentials	Proprietary	Windows	GUI, CLI, Portal, XML-RPC	XenServer, Hyper-V	Citrix Storage Link	No	Yes	Yes	No	Yes	Yes	Yes
Enomaly ECP	GPL v3	Linux	Portal, WS	Xen	No	Amazon EC2	Yes	No	No	No	No	No
Eucalyptus	BSD	Linux	EC2 WS, CLI	Xen, KVM	No	EC2	Yes	No	No	No	No	No
Nimbus	Apache v2	Linux	EC2 WS, WSRF, CLI	Xen, KVM	No	EC2	Yes	Via integration with OpenNebula	Yes (via integration with OpenNebula)	No	No	No
OpenNEbula	Apache v2	Linux	XML-RPC, CLI, Java	Xen, KVM	No	Amazon EC2, Elastic Hosts	Yes	Yes	Yes (via Haizea)	No	No	No
OpenPEX	GPL v2	Multiplatform (Java)	Portal, WS	XenServer	No	No	No	No	Yes	No	No	No
oVirt	GPL v2	Fedor Linux	Portal	KVM	No	No	No	No	No	No	No	No
Platform ISF	Proprietary	Linux	Portal	Hyper-V, XenServer, VMWare ESX	No	EC2, IBM CoD, HP Enterprise Services	Yes	Yes	Yes	Unclear	Unclear	Unclear
Platform VMO	Proprietary	Linux, Windows	Portal	XenServer	No	No	Yes	Yes	No	Yes	No	No
VMWare vSphere	Proprietary	Linux, Windows	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vStorage VMFS	VMware vCloud partners	Yes	VMware DRM	No	Yes	Yes	Yes

AWS infrastructure services



Amazon Web Services (AWS)

IaaS that AWS offer:

- Storage as a Service
 - Amazon Simple Storage Service
 - Amazon SimpleDB
 - Amazon Relational Database Service
- Compute as a Service
 - Amazon Elastic Compute Cloud (EC2)

Different types of Data

- Enterprises have varied requirements for data, **including structured data in relational databases** that power an e-commerce business, or **documents that capture unstructured data** about business processes, plans and visions.
- Enterprises may also need to store objects on behalf of their customers, like an **online photo album or a collaborative document** editing platform.
- Further, some of the data may be **confidential and must be protected**, while others data should be easily shareable.
- In all cases, **business critical data should be secure and available** on demand in the face of hardware and software failures, network partitions and inevitable user errors.

Storage as a Service: Amazon Storage Services

- **Simple Storage Service (S3)**: An object store
- **SimpleDB**: A Key-value store
- **Relational Database Service (RDS)**: MySQL instance
and so on.

Amazon's Simple Storage Server (S3)

- Amazon S3 is a **highly reliable, highly available, scalable** and **fast storage** in the cloud for storing and retrieving large amounts of data just through simple web services.
- S3 is a storage service, several S3 browsers exist that allow users to explore their S3 account as if it were a directory (or a folder). There are also file system implementations that let users treat their S3 account as just another directory on their local disk.

S3 Access Methods:

- AWS Console
- Amazon's RESTful API
- SDKs for Ruby and other languages

Amazon S3: How it works?.



Ref: <https://aws.amazon.com/s3/>

Organizing Data In S3: Buckets, Objects and Keys

- **Files** are called **objects** in S3.
- Objects are referred to with **keys** – basically an optional **directory path name** followed by the **name** of the object.
- Objects **in S3 are replicated across multiple geographic locations** to make it resilient to several types of failures (however, consistency across replicas is not guaranteed).
- If **object versioning is enabled**, recovery from inadvertent deletions and modifications is possible.
- S3 objects can be up to **5 Terabytes in size** and there are no limits on the number of objects that can be stored.
- All objects in S3 **must be stored in a bucket**.

Organizing Data In S3: Buckets, Objects and Keys (Cont...)

- Buckets provide a way to keep related objects in one place and separate them from others. **There can be up to 100 buckets** per account and an unlimited number of objects in a bucket.

S3 objects

- Each object has a **key**, which can be used as the path to the resource in an HTTP URL.
- For example, if the bucket is named **johndoe** and the key to an object is **resume.doc**, then its HTTP URL is
http://s3.amazonaws.com/johndoe/resume.doc

or alternatively, <http://johndoe.s3.amazonaws.com/resume.doc>

- URL needs **authentication parameters**; S3 objects **are private by default** and requests should carry authentication parameters that prove the requester has rights to access the object, unless the object has “Public” permissions.

Note: The bucket namespace is shared; i.e., it is not possible to create a bucket with a name that has already been used by another S3 user.

Amazon S3 security and access management

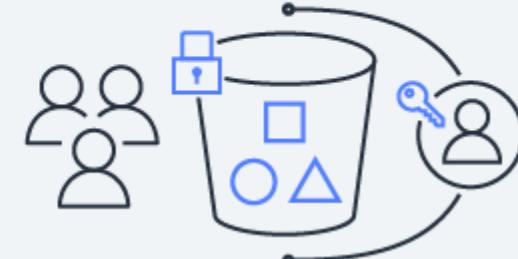
Block Public Access



Object Lock



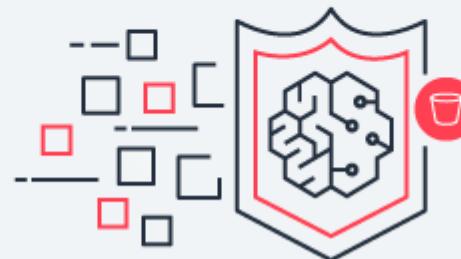
Object Ownership



Identity and Access Management



Amazon Macie

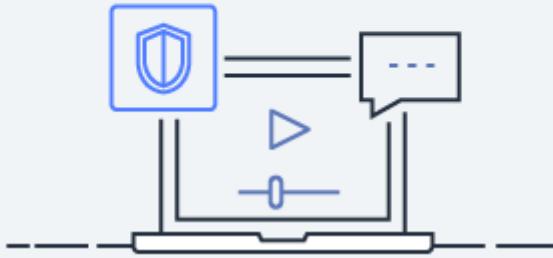


Encryption



Amazon S3 security and access management

AWS Trusted Advisor



AWS PrivateLink for S3



Large Objects and Multi-part Uploads on S3

- The object size limit for S3 is 5 terabytes.
- If this limit is not sufficient, the **object then can be stored in smaller chunks** with the **splitting and re-composition** being managed in the application, using the data.

Note: Uploading large objects on Amazon S3 will still take some time even though it has high aggregate bandwidth available. Additionally, if an upload fails, the entire object needs to be uploaded again.

- Multi-part upload solves both the above problems elegantly.
- S3 provides APIs that allow the developer to write a program that splits a large object into several parts and uploads each part independently.
- These uploads can be parallelized for greater speed to maximize the network utilization.
- If a part fails to upload, only that part needs to be re-tried.

Simple use-case with S3 (Uploading photos)

1. Sign up for S3 at <http://aws.amazon.com/s3/>. While signing up, **obtain the AWS Access Key and the AWS Secret Key**. These are similar to userid and password that is used to authenticate all transactions with Amazon Web Services (not just S3).
2. Sign in to the AWS Management Console for S3 at the below URL
<https://console.aws.amazon.com/s3/home>
3. Create a **bucket** giving a name and geographical location where it can be stored. In S3 all files (called objects) are stored in a bucket, which represents a collection of related objects.
4. Click the Upload button and follow the instructions to upload files.
5. The photos or other files are now safely backed up to S3 and available for sharing with a URL if the right permissions are provided.

Note: From a developer perspective, this can also be accomplished programmatically.

Amazon Simple Database Service (SimpleDB)

- SimpleDB (SDB) provides a simple data store interface in the form of a key-value store.
- It allows storage and retrieval of a set of attributes based on a key.
- It is a highly available, flexible, and scalable non-relational data store that offloads the work of database administration. It provides the core database functions of data indexing and querying in [the cloud](#).
- It provides a simple web services interface to create and store multiple data sets, query your data easily, and return the results.

Data Organization and Access in SimpleDB

- Data is organized into domains.
- Each item in a domain has a unique key that must be provided during creation.
- Each item can have up to 256 attributes, which are name-value pairs.

SDB provides a query language that is analogous to SQL, although there are methods to fetch a single item.

- Queries take advantage of the fact that SDB automatically indexes all attributes.

Availability and Administration in SimpleDB

SDB has a number of features to increase **availability and reliability**.

- Data stored in SDB is automatically replicated across different geographies for high availability.
- It also automatically adds **compute resources in proportion to the request rate** and **automatically indexes all fields** in the dataset for efficient access.
- SDB is **schema-less**; i.e., fields can be added to the dataset as the need arises.

Amazon Relational Database Service (RDS)

- RDS provides a traditional database abstraction in the cloud, specifically a MySQL instance in the cloud.
- An RDS instance can be created using the RDS tab in the AWS Management Console as shown in the next slide.
- AWS performs many of the administrative tasks associated with maintaining a database for the user.
- The database is backed up at configurable intervals, which can be as frequent as 5 minutes.
- The backup data are retained for a configurable period of time which can be up to 8 days. Amazon also provides the capability to snapshot the database as needed.

COMPUTE AS A SERVICE: AMAZON ELASTIC COMPUTE CLOUD (EC2)

Compute as a Service:

- Here, computing resources are offered as a service to the users.
- It should be **possible to associate storage with the computing service** (so that the results of the computation can be made persistent).
- **Virtual networking** is needed as well, so that it is possible to communicate with the computing instance.

All these together make up Infrastructure as a Service.

Note: Amazon's Elastic Compute Cloud (EC2) is one of the popular Compute as a Service offerings

EC2 Computational Resources

Computing Resources:

- The computing resources available on EC2, referred to as **EC2 instances**, consist of combinations of **computing power, together with other resources such as memory**.
- Amazon measures the **computing power** of an EC2 instance in **terms of EC2 Compute Units**.
- An EC2 Compute Unit (CU) is a standard **measure of computing power** in the same way that bytes are a standard measure of storage.
- One EC2 CU provides the same amount of computing power as a 1.0–1.2 GHz Opteron or Xeon processor in 2007

For example, if a developer requests a computing resource of 1 EC2 CU, and the resource is allocated on a 2.4 GHz processor, they may get 50% of the CPU. This allows developers to request standard amounts of CPU power regardless of the physical hardware.

Instance Types in EC2

A developer can request a computing resource of one of the instance types shown in the table.

Instance Type	Compute Capacity	Memory	Local Storage	Platform
Small	1 virtual core of 1 CU	1.7GB	160GB	32-bit
Large	2 virtual cores, 2 CU each	7.5GB	850GB	64-bit
Extra Large	4 virtual cores, 2 CU each	15GB	1690GB	64-bit

Note: High memory instances are also available for databases and other memory-hungry applications

EC2 Softwares

- Amazon makes available operating system and application software in the form of **Amazon Machine Images(AMIs)**.
- Operating systems available in AMIs include various flavors of **Linux**, such as **Red Hat Enterprise Linux and SuSE**, the **Windows server**, and **Solaris**.
- The required AMI has to be specified **when requesting the EC2 instance**, as demonstrated. The AMI running on an EC2 instance is also called the **root AMI**.
- Software available includes databases such as IBM DB2, Oracle and Microsoft SQL Server. A wide variety of other application software and middleware, such as Hadoop, Apache, and Ruby on Rails, are also available.

Accessing additional softwares on EC2

There are two ways of using additional software not available in standard AMIs.

1. Request a standard AMI, and then install the additional software needed.
This AMI can then be saved as one of the available AMIs in Amazon.
2. Import a VMware image as an AMI using the **ec2-import-instance** and **ec2-import-disk-image** commands

Regions and Availability Zones of EC2

- EC2 offers **regions**, which are the same as the S3 regions described in the earlier slides of S3.
- Within a **region**, there are multiple **availability zones**, where each availability zone corresponds to a virtual data center that is isolated (for failure purposes) from other availability zones.

For example, an enterprise that wishes to have its EC2 computing instances in Europe could select the “Europe” region when creating EC2 instances.

- By creating two instances in different availability zones, the enterprise could have a highly available configuration that is tolerant to failures in any one availability zone.

Load Balancing and Scaling on EC2

- EC2 provides the **Elastic Load Balancer**, which is a service that balances the load across multiple servers.
- The default load balancing policy is to treat all requests as being independent.
- It is also possible to have **timer-based and application controlled sessions**, whereby successive requests from the same client are routed to the same server based upon time or application direction.
- The **load balancer** also scales the number of servers up or down depending upon the load. This can also be used as a failover policy, since failure of a server is detected by the Elastic Load Balancer.
- If the load on the remaining server is too high, the **Elastic Load Balancer** could start a new server instance.

EC2 Storage Resources

1. **Amazon S3:** Highly available object store (already covered in the earlier slides).
2. **Elastic Block Service:** Permanent block storage
3. **Instance Storage:** Transient block storage.

2. Elastic Block Service (EBS)

- In the same way that S3 provides file storage services, EBS provides a block storage service for EC2.
- It is possible to request an EBS disk volume of a particular size and attach this volume to one or multiple EC2 instances using the instance ID returned during the time the volume is created.
- The EBS volume has an existence independent of any EC2 instance, which is critical to have persistence of data

3. Instance Service

- Every EC2 instance has local storage that can be configured as a part of the compute resource. This is referred to as instance storage.
- Storage exists only as long as the EC2 instance exists, and cannot be attached to any other EC2 instance.
- If the EC2 instance is terminated, the instance storage ceases to exist.
- To overcome this limitation of local storage of Instance service, developers can use either **EBS or S3** for persistent storage and sharing.

Comparison of Instance Storage and EBS Storage

	Instance Storage	EBS storage
Creation	Created by default when an EC2 instance is created	Created independently of EC2 instances.
Sharing	Can be attached only to EC2 instance with which it is created.	Can be shared between EC2 instances.
Attachment	Attached by default to S3-backed instances; can be attached to EBS-backed instances	Not attached by default to any instance.
Persistence	Not persistent; vanishes if EC2 instance is terminated	Persistent even if EC2 instance is terminated.
S3 snapshot	Can be snapshotted to S3	Can be snapshotted to S3

EC2 Networking Resources

- Network resources are also needed by applications in addition to compute and storage resources.
- For networking between EC2 instances, EC2 offers both a **public address** as well as a **private address** to each instance.
- It also offers **DNS services** for managing DNS names associated with these IP addressees. **Amazon Route 53** effectively connects user requests to infrastructure running in AWS – such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets – and can also be used to route users to infrastructure outside of AWS.
- **Elastic IP address:** It is a public IPv4 address, which is reachable from the internet. These addresses are independent of any instance, and can be used to support failover of servers

Virtual Private Cloud

- A virtual private cloud (VPC) is a secure, isolated **private cloud** hosted within a **public cloud**.
- Enterprises that desire more control over their networking configuration can use Virtual Private Cloud (VPC). Examples of VPC are provided below:
 1. The ability to allocate both public and private IP addresses to instances from any address range.
 2. The ability to divide the addresses into subnets and control the routing between subnets.
 3. The ability to connect the EC2 network with an Intranet using a VPN tunnel.

Simple EC2 Example: Setting up a Web Server

The web server will be created as an EBS-backed instance, to avoid the necessity of having to periodically back up the storage to S3.

The process is broken down into four steps:

1. Selecting the AMI for the instance (Amazon Images" and "Amazon Linux" brings up a list of Linux images supplied by Amazon).
2. Creating the EC2 instance and installing the web server.
3. Creating an EBS volume for data, such as HTML files and so on.
4. Setting up networking and access rules (for allowing external access to the Web server).

Assumptions:

- i. The data needed for the web server (HTML files, scripts, executables, and so on) are available, and have been uploaded to EC2.
- ii. The web server needed also has to be uploaded to EC2 and then installed (in reality, a web server instance may be available as an image as well)

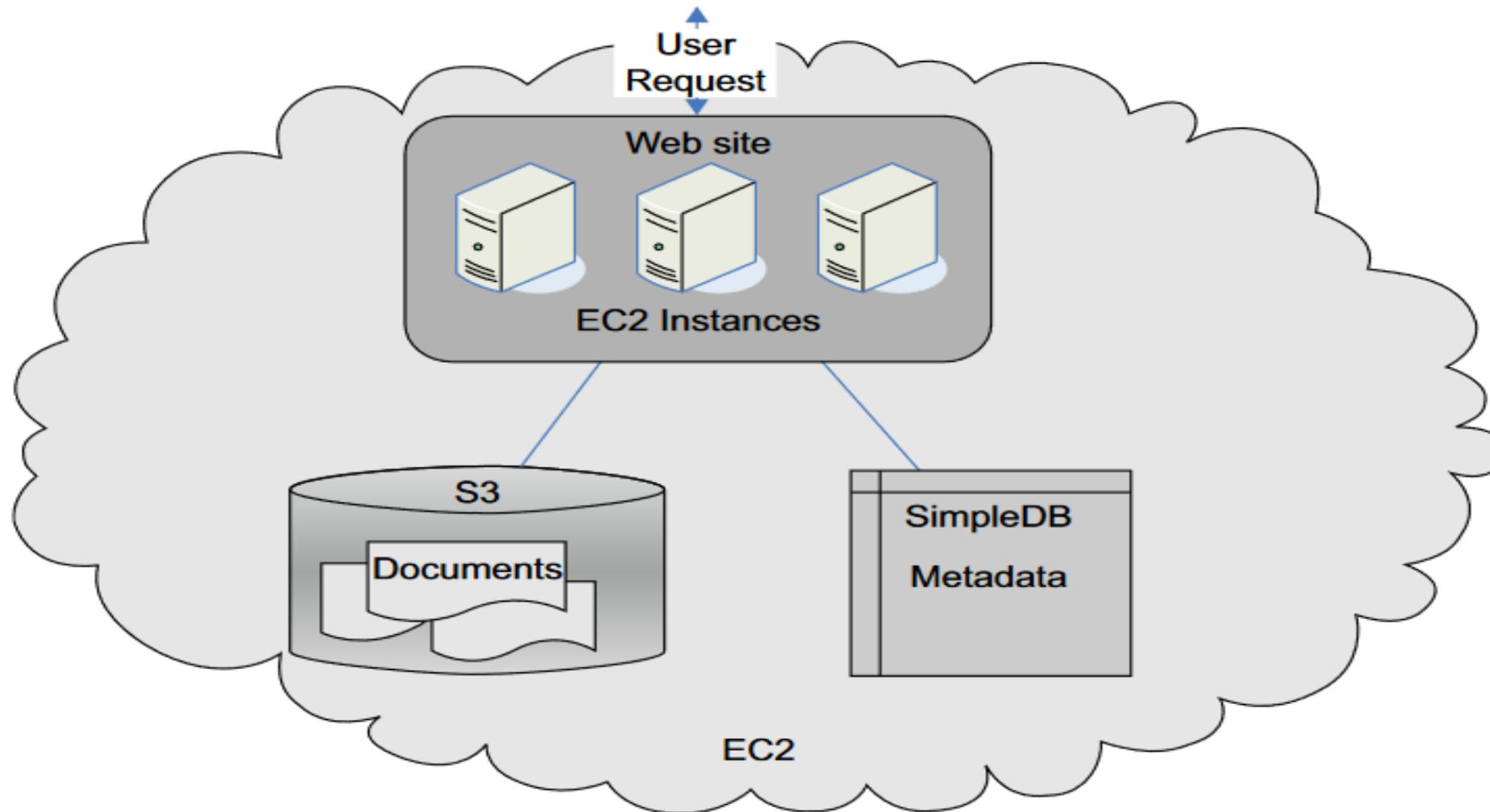
Case study of Using Amazon EC2 for Pustak Portal

What is Pustak Portal (a simple book publishing portal): It allows authors to upload and share book chapters or short articles in various formats with readers, who have to be registered with the portal.

Requirements:

- It is necessary to store the documents, together with metadata such as the file type, and an access control list of readers who have been given access permission.
- As a particular article may become very popular due to its topical nature, the load on the portal could vary greatly, and it is necessary that the number of servers scale up and down with usage.

The High-level architecture of the enhanced Pustak Portal

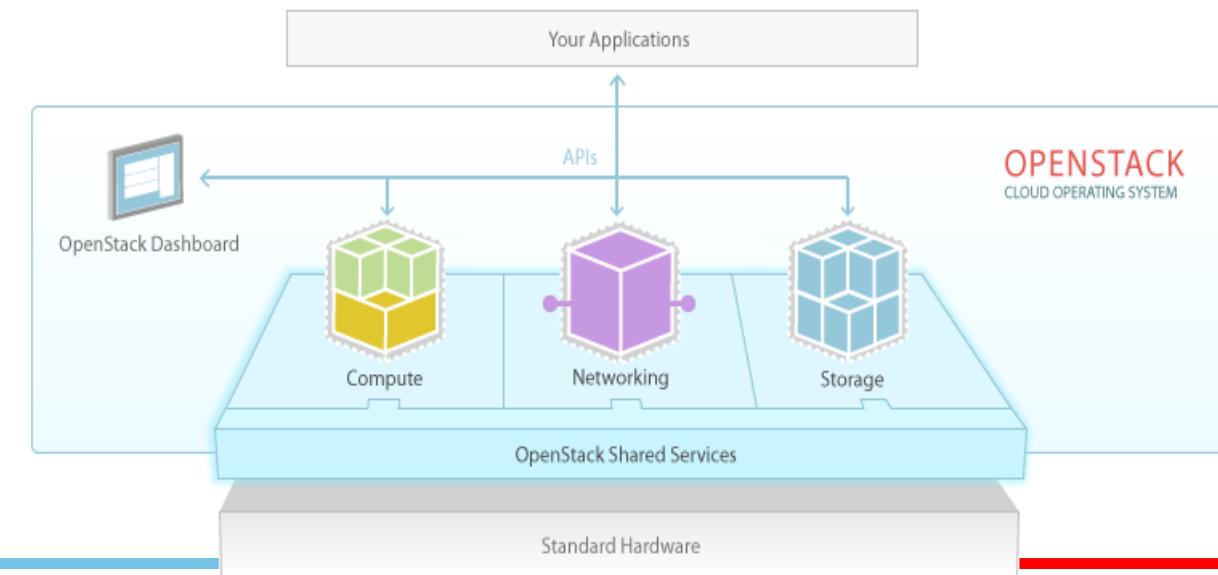


Pustak Portal (Cont...)

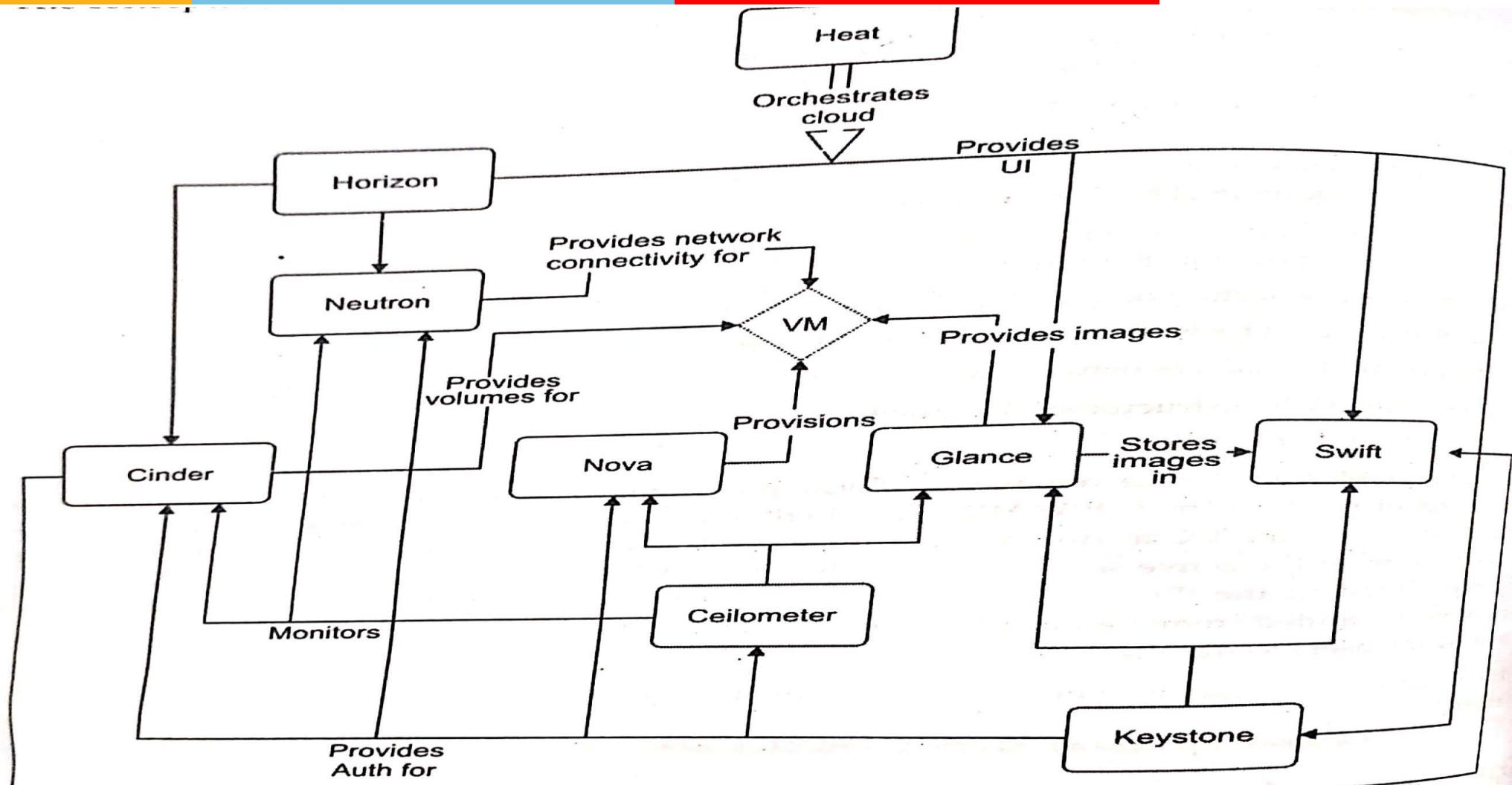
- Amazon S3 is used for storing articles.
 - Read object
 - Write object
 - Delete object
- The associated metadata of articles, such as the name of the article, author, and a list of readers, etc., are stored in SimpleDB.
 - Connect to database
 - Read data
 - Write data
 - Search database

Topic: Openstack

- OpenStack is considered as – Infrastructure as a Service (IaaS).
- It is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.
- It is one among several open-source cloud building software through which various organizations offer cloud services to their clients.
- The cloud can run on the commodity hardware that are available at economical costs.



Conceptual OpenStack Architecture



Logical OpenStack Architecture

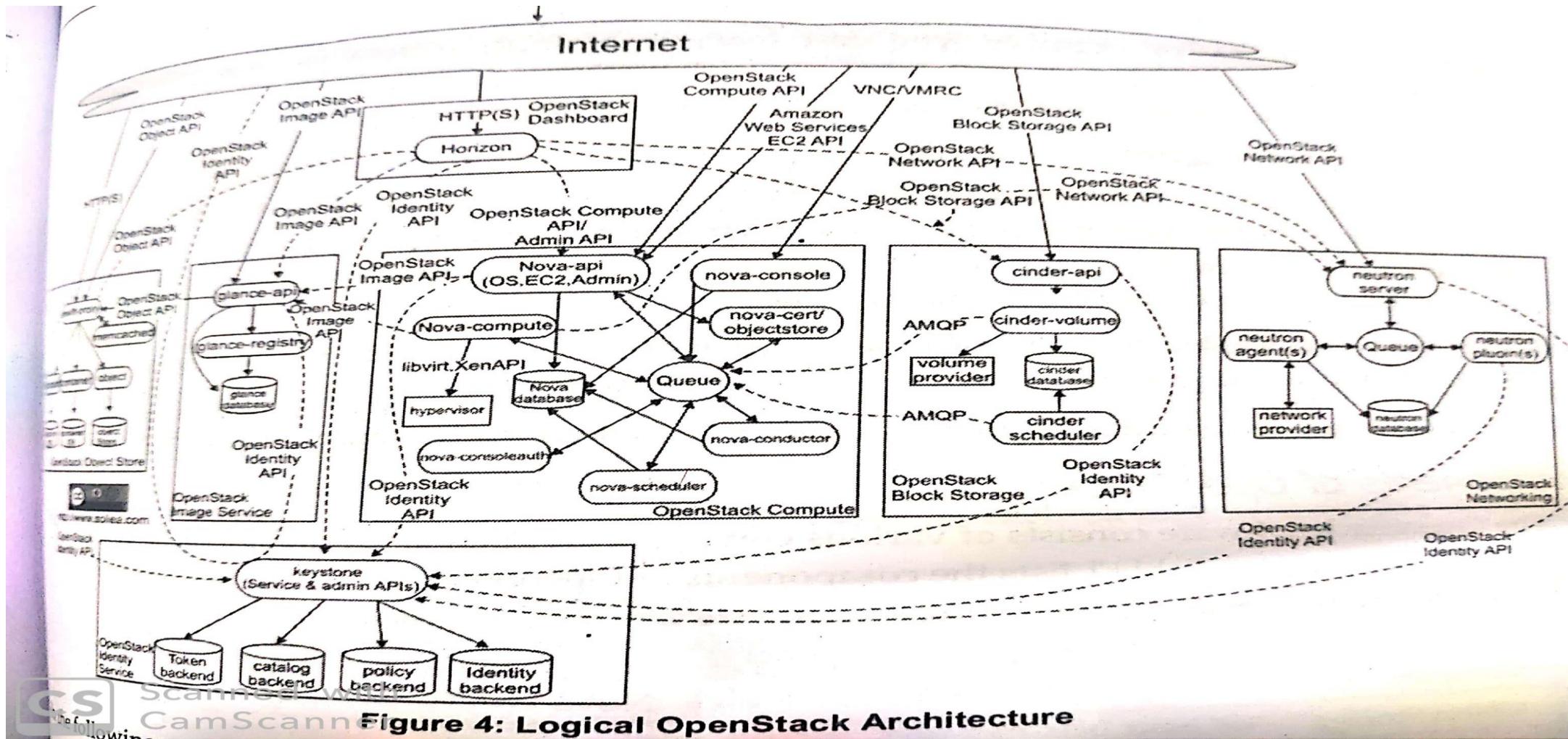
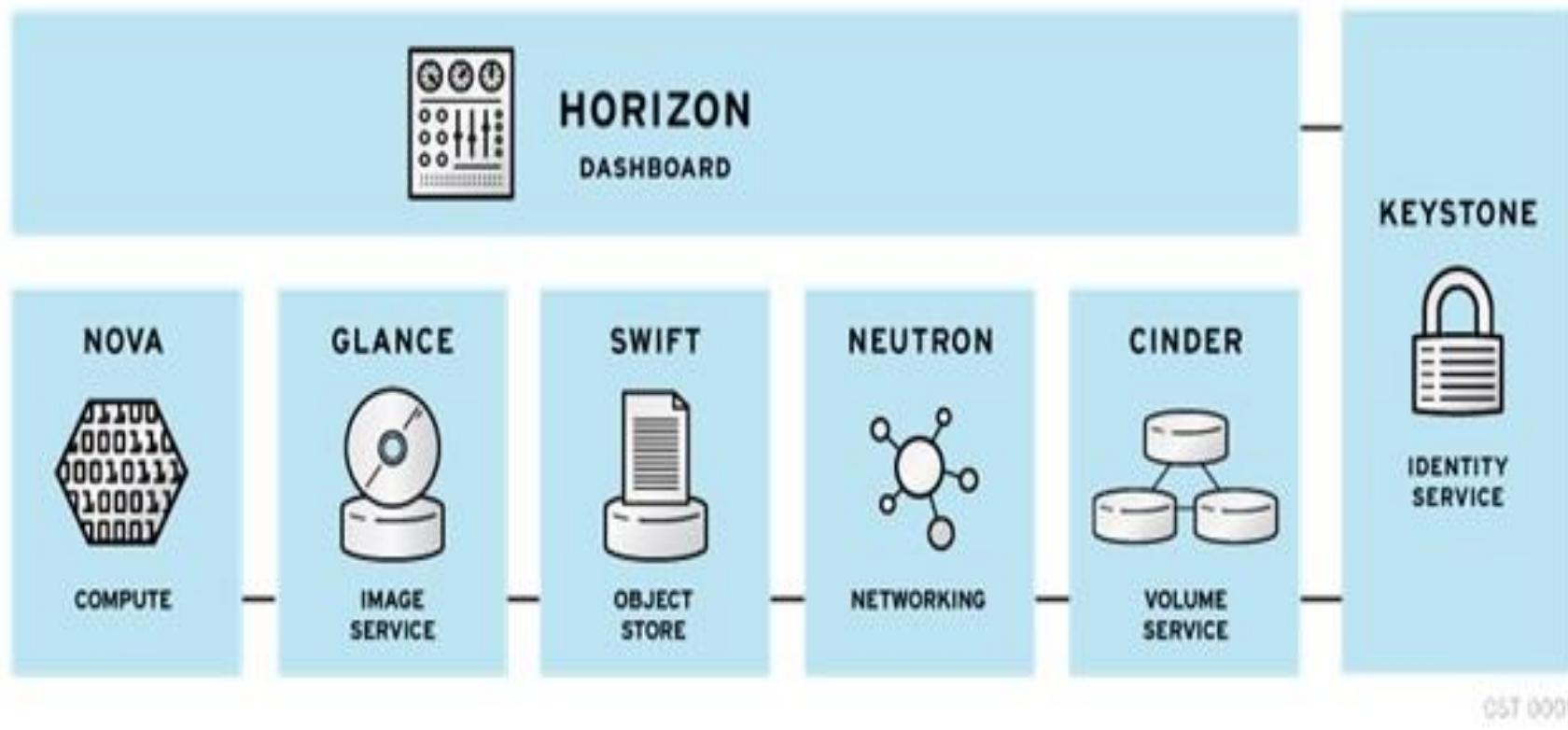


Figure 4: Logical OpenStack Architecture

Openstack Key Components



Openstack Components. Cont.

Horizon – Dashboard

- It provides a modular web-based user interface for all the OpenStack services. With this web GUI, you can perform most operations on your cloud like **launching an instance, assigning IP addresses and setting access controls**.
- Its primary objective is to interact with the backend API's of other components and execute requests initiated by users.
- It interacts with keystone authentication service, to authorize requests before doing anything

Project



Admin / Overview

Admin



Overview

Overview

Compute



Volume



Usage Summary

Network



Select a period of time to query its usage:

System



The date should be in YYYY-MM-DD format.

Identity



2018-01-06 to 2018-01-07

Submit

Active Instances: 0

Active RAM: 0Bytes

This Period's VCPU-Hours: 3.85

This Period's GB-Hours: 0.01

This Period's RAM-Hours: 4120.75

Usage

Download CSV Summary

Displaying 55 items

Project Name	VCPUs	Disk	RAM	VCPU Hours <small>?</small>	Disk GB Hours <small>?</small>	Memory MB Hours <small>?</small>
1e0b0f99af0249cea24502034d73d356 (Deleted)	0	0Bytes	0Bytes	0.01	0.00	7.96
bf11c5fa0f1b452db4b7741a6c33a92f (Deleted)	0	0Bytes	0Bytes	0.49	0.00	497.78
ea1f2f357c09465eb6991edf7079efbe (Deleted)	0	0Bytes	0Bytes	0.11	0.00	110.93

Activate Windows

Go to Settings to activate Windows.

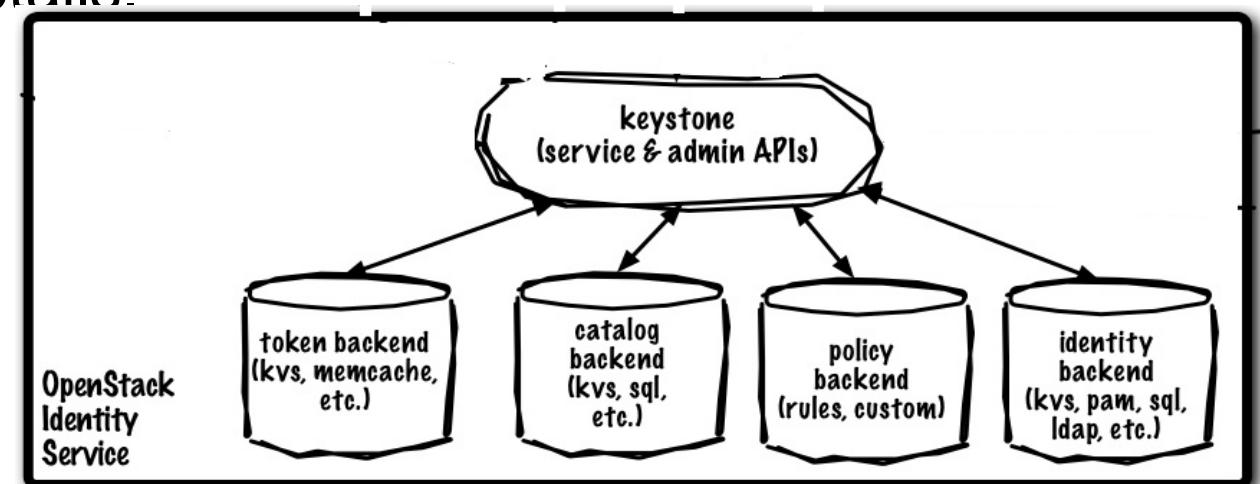
OpenStack
dashboard
Admin tab

Keystone – Identity

Keystone is a framework for authentication and authorization for all the OpenStack services.

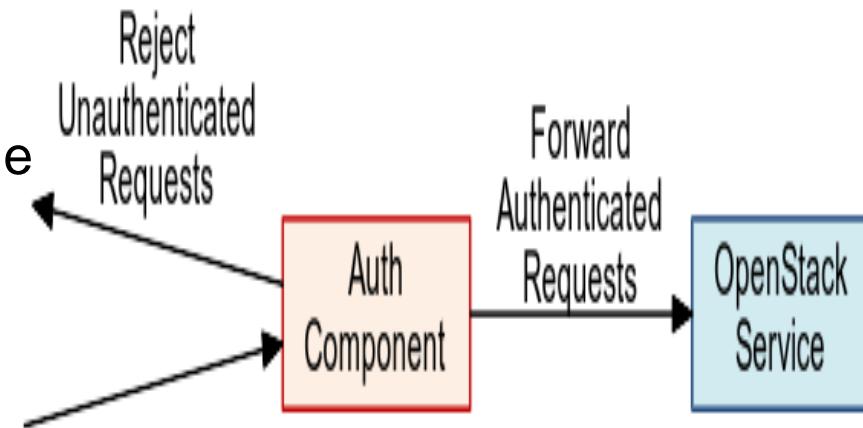
Keystone has two primary functions

- 1) Manage Users. Like tracking of all users, and their permissions.
- 2) Service list/catalog. This is nothing but providing information regarding what services are available and their respective API endpoint details.



Keystone (OpenStack Identity Service):

The OpenStack Identity Service provides the cloud environment with an authentication and authorization system. In this system, users are a part of one or more projects. In each of these projects, they hold a specific role. Users need to have identity and a particular level of access in the cloud. When a user logs into the cloud, Keystone authenticates that he is indeed a user and authorises his level of access within the cloud.



Glance – Image Store

It provides discovery, registration and delivery services for disk and server images.

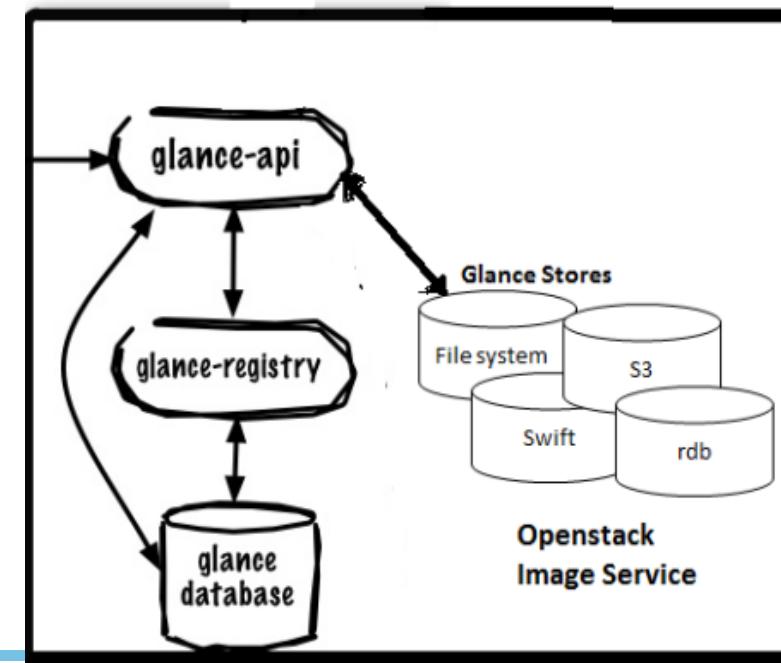
List of processes and their functions:

glance-api : It accepts Image API calls for image discovery, image retrieval and image storage.

glance-registry : it stores, processes and retrieves metadata about images (size, type, etc.).

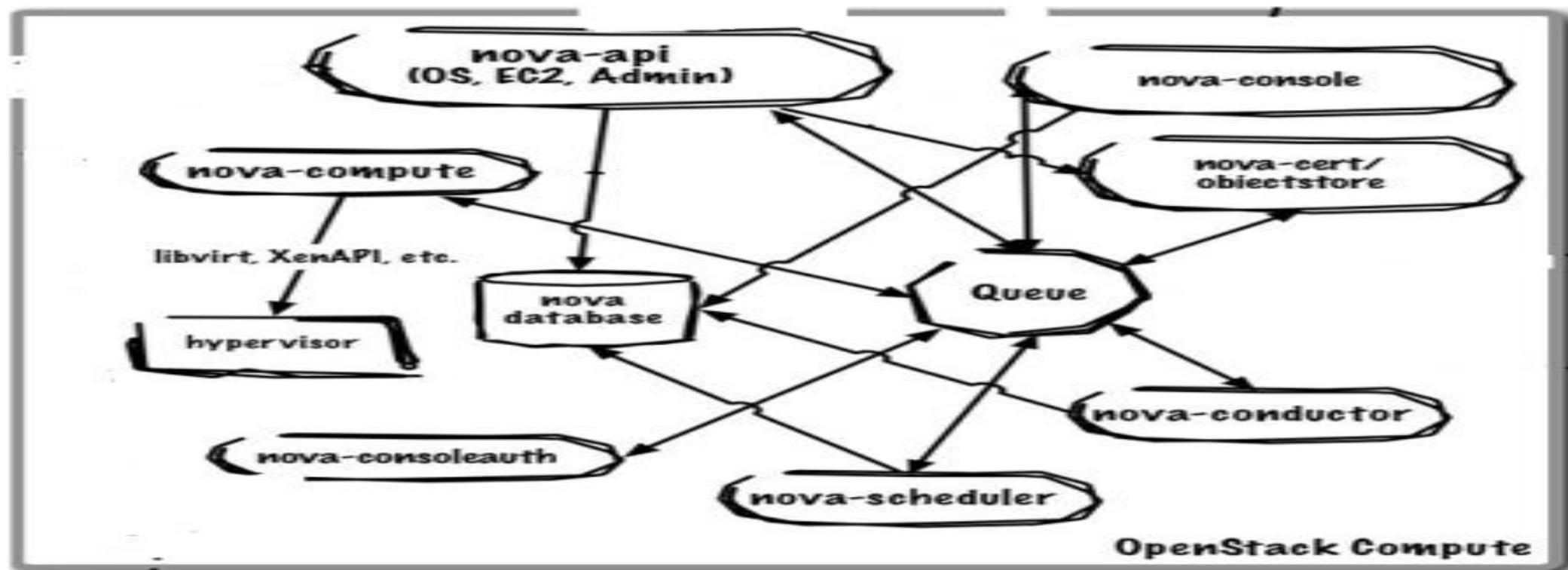
glance database : A database to store the image metadata.

A ***storage repository*** for the actual image files. Glance supports normal file-systems, Amazon S3, and Swift.



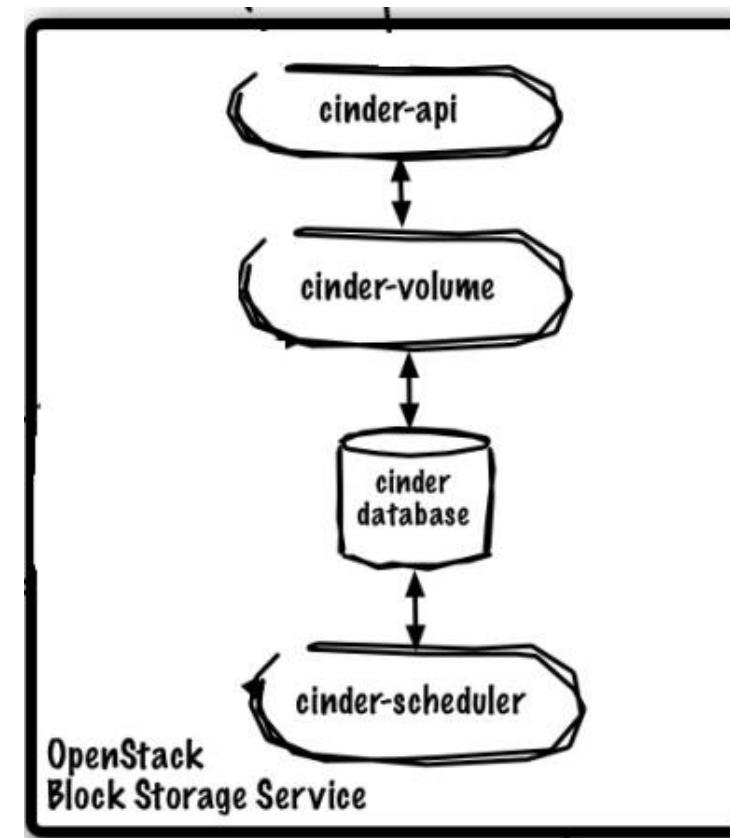
Nova – Compute

It provides virtual servers upon demand. Nova is the most complicated and distributed component of OpenStack. A large number of processes cooperate to turn end user API requests into running virtual machines.



Cinder – Block Storage

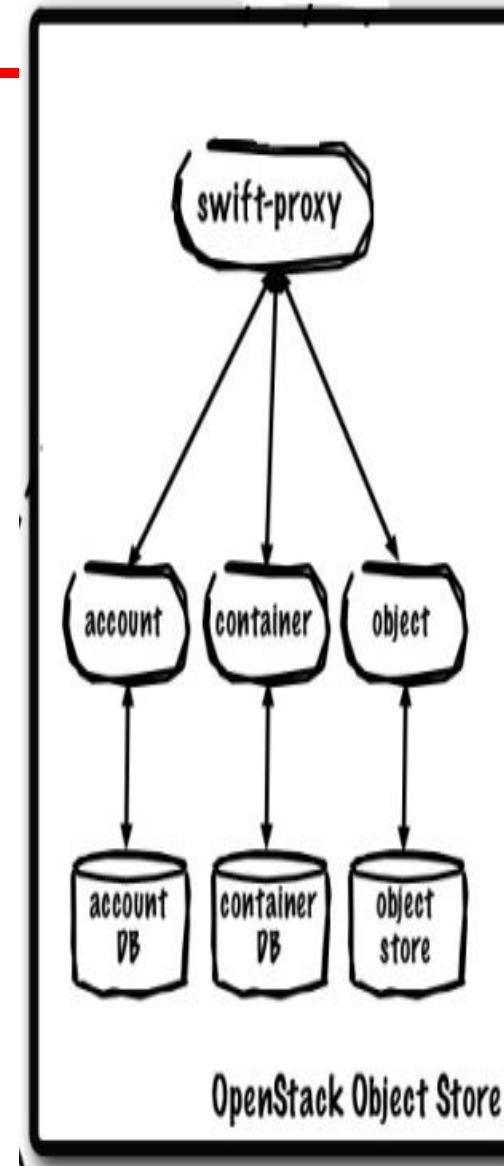
Cinder allows block devices to be exposed and connected to compute instances for expanded storage & better performance.



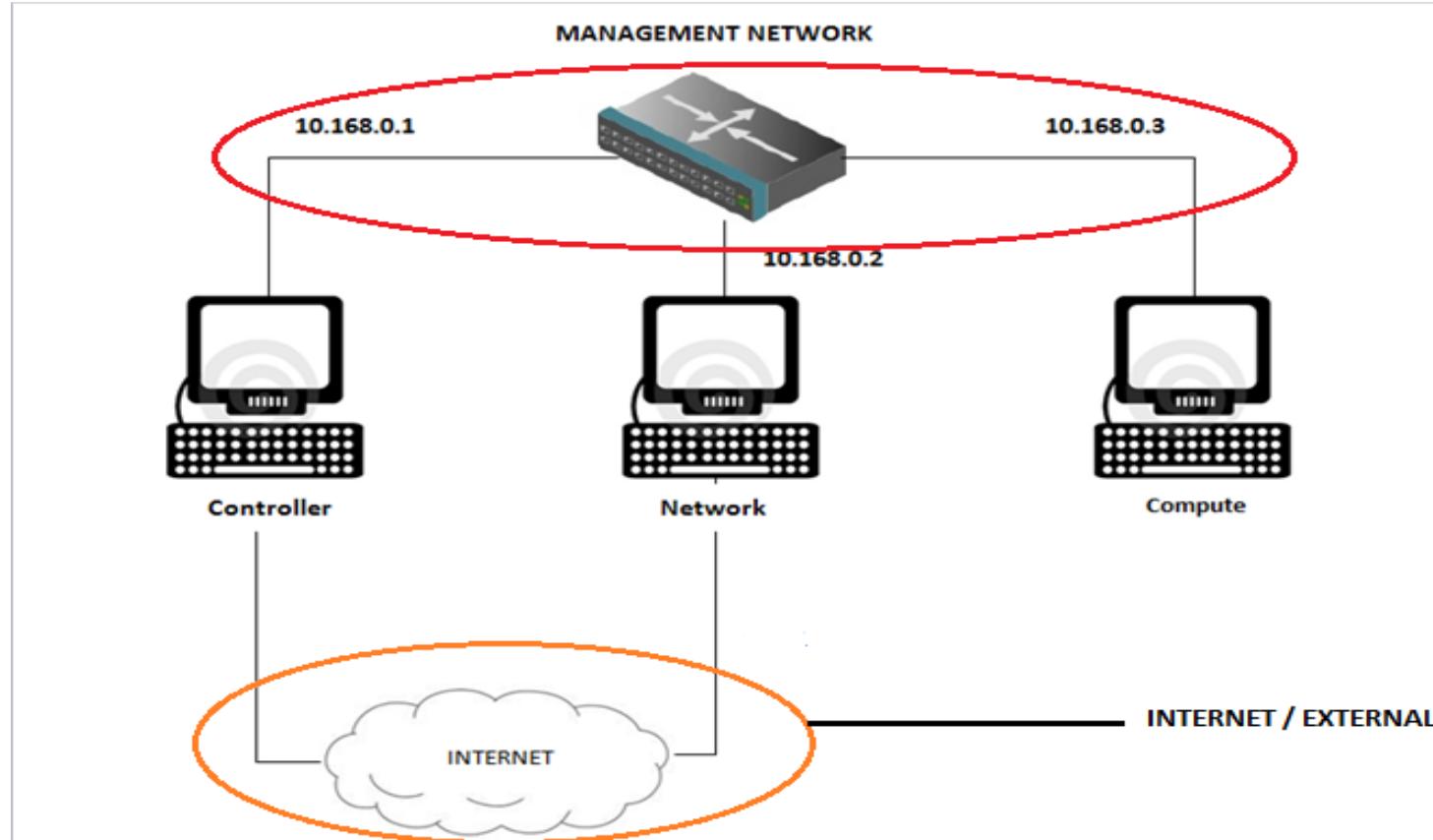
Swift – Object Storage

Object store allows you to store or retrieve files. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention.

Note : Object Storage is not a traditional file system, but rather a distributed storage system for static data such as virtual machine images, photo storage, email storage, backups and archives.



Networking



There are two networks :

1. Internal or Management network
2. External or Internet network

CONTROLLER NODE

- The controller is the central management system in a multinode cloud installation.
- The Controller node supplies API, scheduling, and other shared services for the cloud.
- The Controller node has the dashboard, the image store, and the identity service. Additionally, Nova compute management service as well as the Neutron server are also configured in this node.

Component Name	Used for	Similar to
Horizon	A dashboard for end users or administrators to access other backend services	AWS Management Web Console
Nova Compute	Manages virtualization and takes requests from end user through dashboard or API to form virtual Instances	AWS Elastic Compute
Cynder	For Block storage, directly attachable to any virtual instance, similar to an external hard drive	EBS(Elastic Block Store)
Glance	This is used for maintaining a catalog for images and is kind of a repository for images.	AMI (Amazon Machine Images)
Swift	This is used for Object storage that can be used by your applications or instances to store static objects like multimedia files, backups, store images, archives etc.	AWS S3
Keystone	This component is responsible for managing authentication services for all components. Like a credentials and authorization, and authentication for users	AWS Identity And Access Management(IAM)

OpenStack Installation References (Taxila)

- https://youtu.be/x5tuyzwq16k?list=PLvvQ7qimT0kmJFGS_uYIOA423PlvVmxOg (Installation)
- https://youtu.be/wzVSGGg4fsY?list=PLvvQ7qimT0kmJFGS_uYIOA423PlvVmxOg (Instance Creation)
- https://youtu.be/G1ZY4RorBiw?list=PLvvQ7qimT0kmJFGS_uYIOA423PlvVmxOg (Instance Creation with Volume)
- https://youtu.be/QFkfSgjJddI?list=PLvvQ7qimT0kmJFGS_uYIOA423PlvVmxOg (Swift Object Storage)
- Cloud Computing Black Book, Kailash Jaiswal, etc., 2020 Edition.

Topic: Managing Virtual Resources on the Cloud: Provisioning and Migration

Public cloud: Public cloud or external cloud describes cloud computing in a traditional mainstream sense, whereby resources are dynamically provisioned via publicly accessible Web applications/Web services (SOAP or RESTful interfaces) from an off-site third-party provider, who shares resources and bills on a fine-grained utility computing basis, the user pays only for the capacity of the provisioned resources at a particular time.

There are many examples for vendors who publicly provide **infrastructure as a service**. **Amazon Elastic Compute Cloud (EC2)** is the best known example. Few other examples **GoGrid**, **Joyent Accelerator**, **Rackspace**, **AppNexus**, **FlexiScale**, and **ManjrasoftAneka**.

Topic: Managing Virtual Resources on the Cloud: Provisioning and Migration (Cont...)

Private Cloud: A private cloud aims at providing public cloud functionality, but on private resources, **while maintaining control over an organization's data and resources to meet security and governance's requirements** in an organization. Private cloud exhibits a highly virtualized cloud data center **located inside your organization's firewall.**

- The best-known examples are **Eucalyptus** and **OpenNebula**.

High Availability: It allows virtual machines to automatically be restarted in case of an underlying hardware failure or individual VM failure.

- If one of your servers fails, the VMs will be restarted on other virtualized servers in the resource pool, restoring the essential services with minimal service interruption.

Two core services of getting resources for VMs

In Infrastructure as a Service (IaaS), the provisioning of required resources for systems and applications on a large number of physical machines is traditionally a time-consuming process with low assurance on deployment's time and cost.

Two core services are there that enable the users to get the best out of the IaaS model in public and private cloud setups.

- 1) Virtual machine provisioning and**
 - 2) Migration services**
-

Why Virtual Machine Provisioning is required?

To provide a new virtual machine in a matter of minutes, **saving lots of time and effort.**

Generic procedure for virtual machine provisioning:

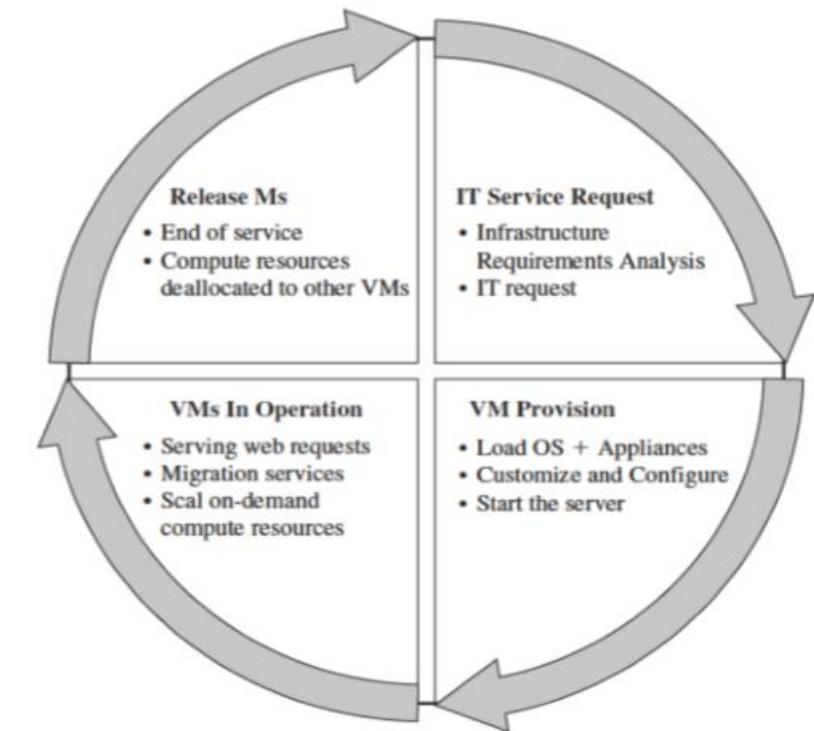
-Check the inventory for a new machine, get one, format, install OS required, and install services; a server is needed along with lots of security batches and appliances.

With the emergence of virtualization technology, **it is just a matter of minutes to achieve the above task of IT admin through** public cloud virtualization management software package or a private cloud management solution installed at your data center in order to provision the virtual machine inside the organization and within the private cloud setup.

Virtual Machine Provisioning and Management Life Cycle

- The cycle starts by a request delivered to the IT department, stating the requirement for creating a new server for a particular service.
- This request is being processed by the IT administration to start seeing the servers' resource pool, matching these resources with requirements
- Starting the provision of the needed virtual machine.
- Once it provisioned and started, it is ready to provide the required service according to an SLA(Service Level agreement).
- Virtual is being released; and free resources.

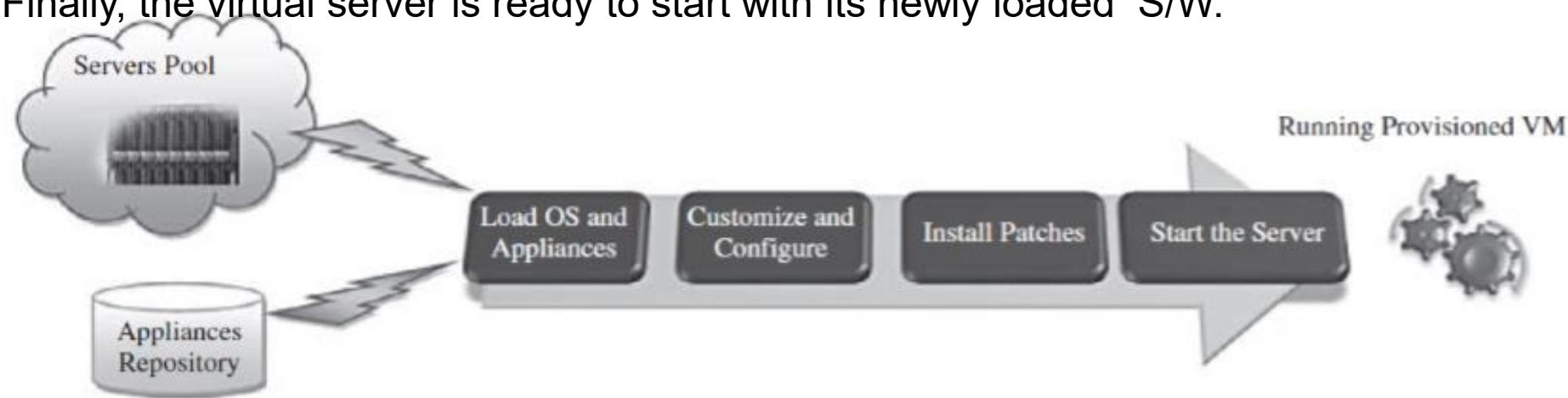
Virtual Machine Life Cycle



VM Provisioning Process

Steps to Provision VM -

- Select a server from a pool of available servers along with the appropriate OS template you need to provision the virtual machine.
 - Load the appropriate software (operating system, device drivers, middleware, and the needed applications for the service) .
 - Customize and configure the machine (e.g., IP address, Gateway) to an associated network and storage resources.
 - Finally, the virtual server is ready to start with its newly loaded S/W.



VM Provisioning using templates

- Provisioning from a template reduces the time required to create a new virtual machine.
- Administrators can create different templates for different purposes.

For example –

- **Vagrant provision** tool using VagrantFile (template file) (demo)
- **Heat** – Orchestration Tool of openstack (Heat template in YAML format)
(demo – Instance creation in cloud, Load balancer in cloud)

This enables the administrator to quickly provision a correctly configured virtual server on demand.

Why Migration is required?

Migrations of a virtual machine is a matter of milliseconds: **saving time, effort, making the service alive for customers, and achieving the SLA/SLO agreements and quality-of-service (QoS) specifications required.**

- A particular VM is consuming more than its fair share of resources at the expense of other VMs on the same host, it will be eligible, for this machine, to either be moved to another underutilized host or assign more resources for it.

Virtual Machine Migration Services

Migration service -

The process of moving a virtual machine from one host server or storage location to another. It plays an important role in datacenters by making it easy to adjust resource's priorities to match resource's demand conditions.

There are different techniques of VM migration-

- **Cold/regular migration,**
- **Hot/live migration, and**
- **Live storage migration of a virtual machine.**

In this process, all key machines' components, such as **CPU, storage disks, networking, and memory**, are completely virtualized, thereby facilitating the entire state of a virtual machine to be captured by a set of easily moved data files.

Cold/regular migration

Cold migration is the migration of a **powered-off virtual machine** and is done in the following tasks:

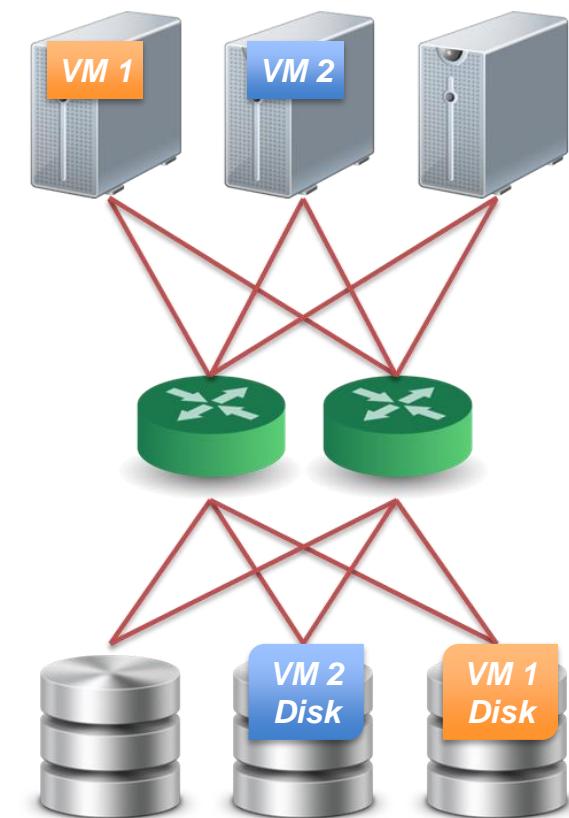
- If the option to move to a different datastore was chosen, the configuration files, including the **NVRAM file (BIOS settings)**, **and log files** are moved from the source host to the destination host's associated storage area. If you chose to move the virtual machine's disks, these are also moved.
- The virtual machine is registered with the new host.
- After the migration is completed, the old version of the virtual machine is deleted from the source host if the option to move to a different datastore was chosen.

Live Migration Technique (hot or real-time migration)

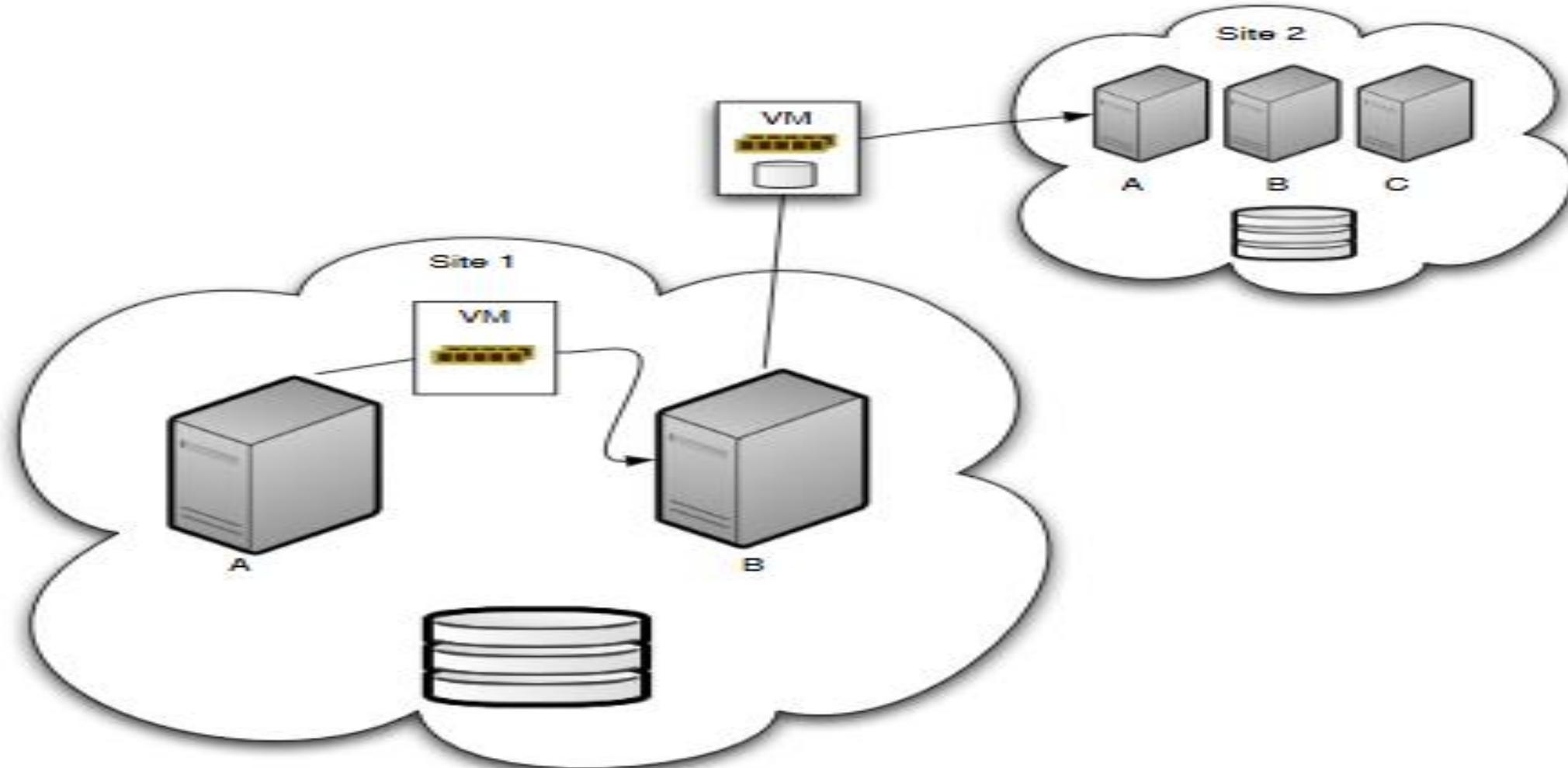
It can be defined as the movement of a virtual machine from one physical host to another while being powered on. When it is properly carried out, this process takes place without any noticeable effect from the end user's point of view (a matter of milliseconds). Live migration can also be used for load balancing.

Pre-assumption :

- We assume that all storage resources are separated from computing resources.
- Storage devices of VMs are attached from network :
 - **NAS**: NFS, CIFS
 - **SAN**: Fibre Channel
 - **iSCSI**, network block device
 - **drdb** network RAID
- Require high quality network connection
 - Common L2 network (LAN)
 - L3 re-routing



In-site and Cross-site Migration



Live Migration Technique

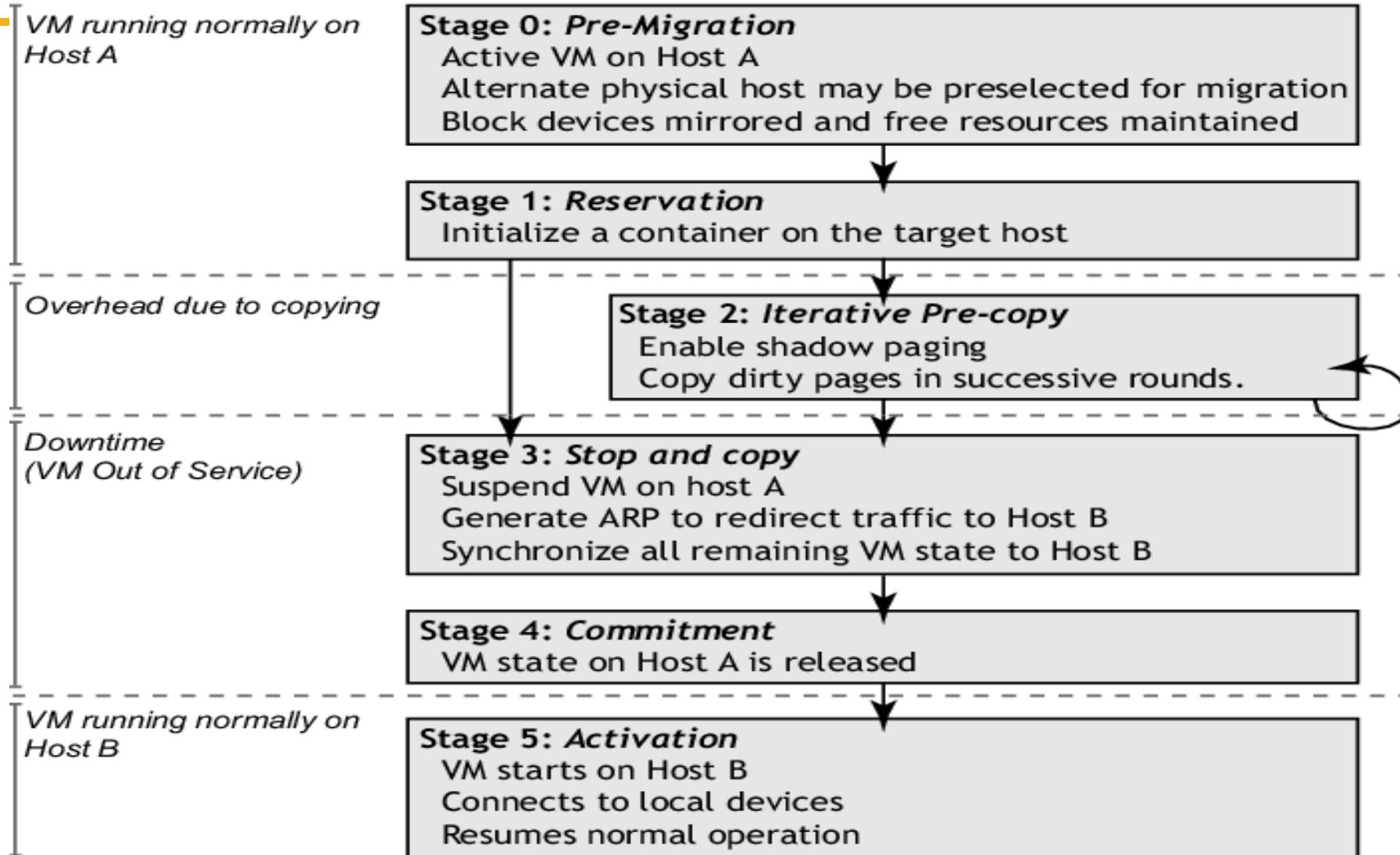
Challenges of live migration :

- VMs have lots of state in memory
- Some VMs have soft real-time requirements :
 - For examples, web servers, databases and game servers, ...etc.
 - Need to minimize down-time

Relocation strategy :

1. Pre-migration process
2. Reservation process
3. Iterative pre-copy
4. Stop and copy
5. Commitment

Live Migration Technique

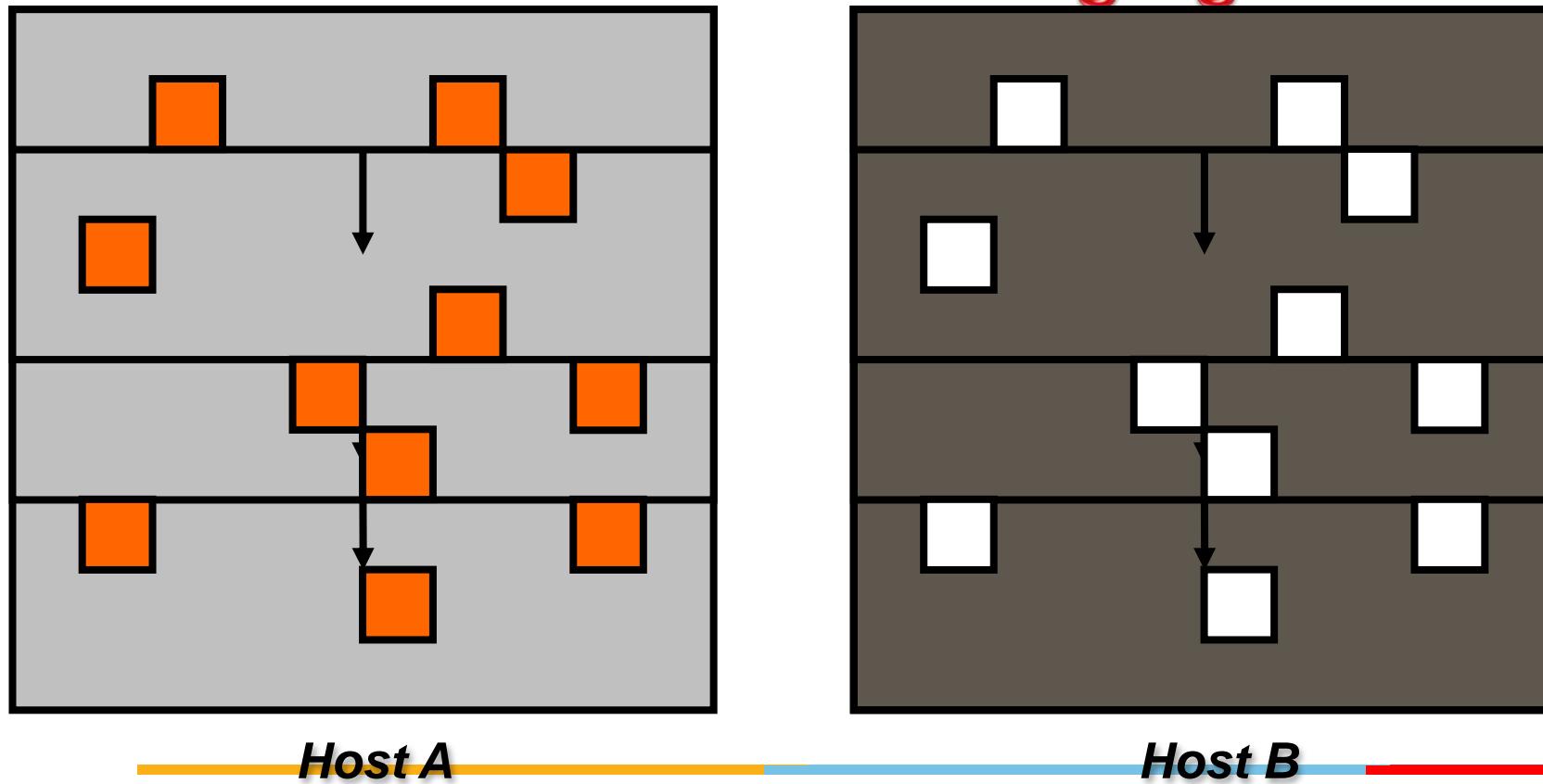


Note: We can migrate but with Short downtime.

Live Migration Technique

Live migration process :

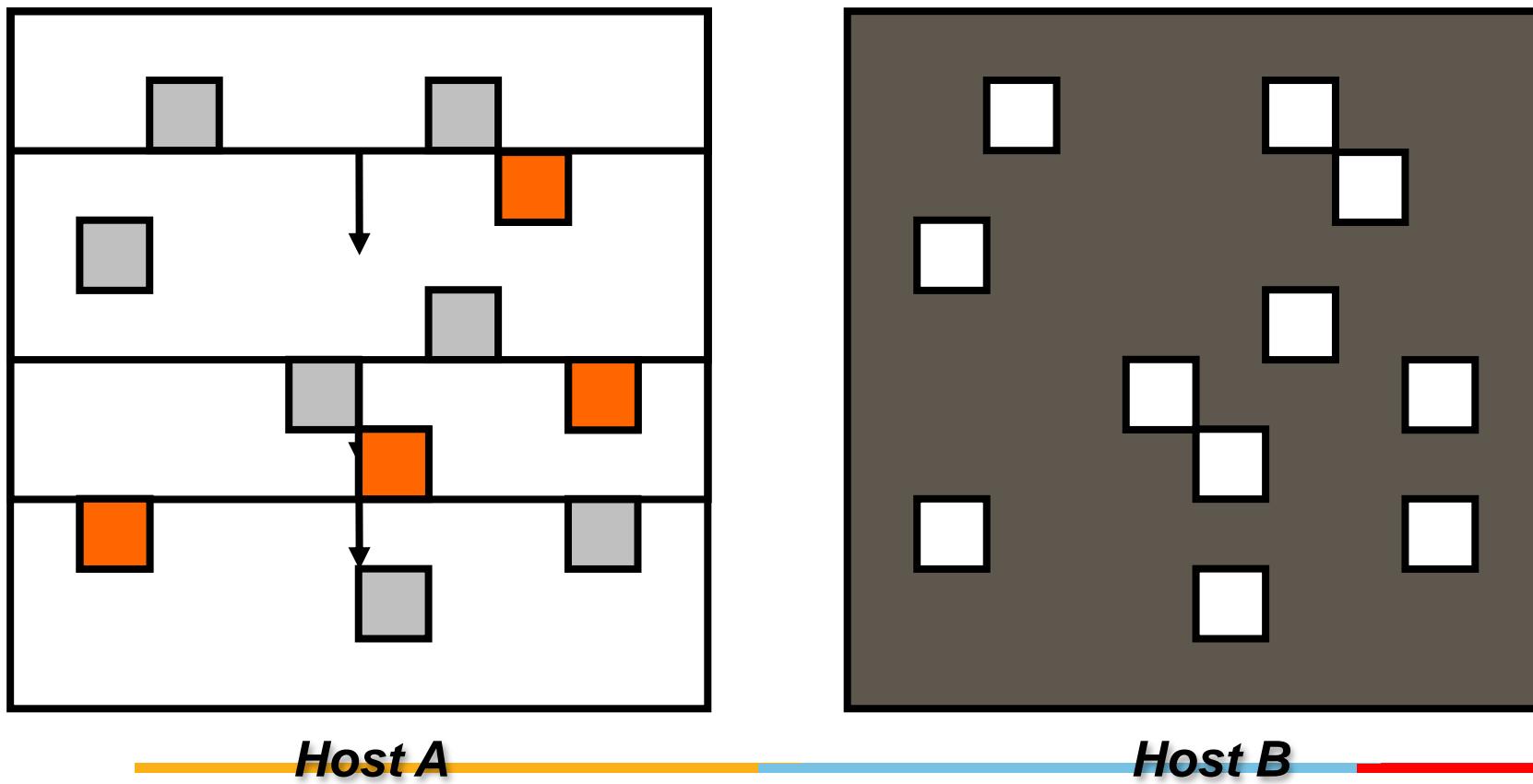
**Pre-copy migration : Round 1,
Enable Shadow Paging**



Live Migration Technique

Live migration process :

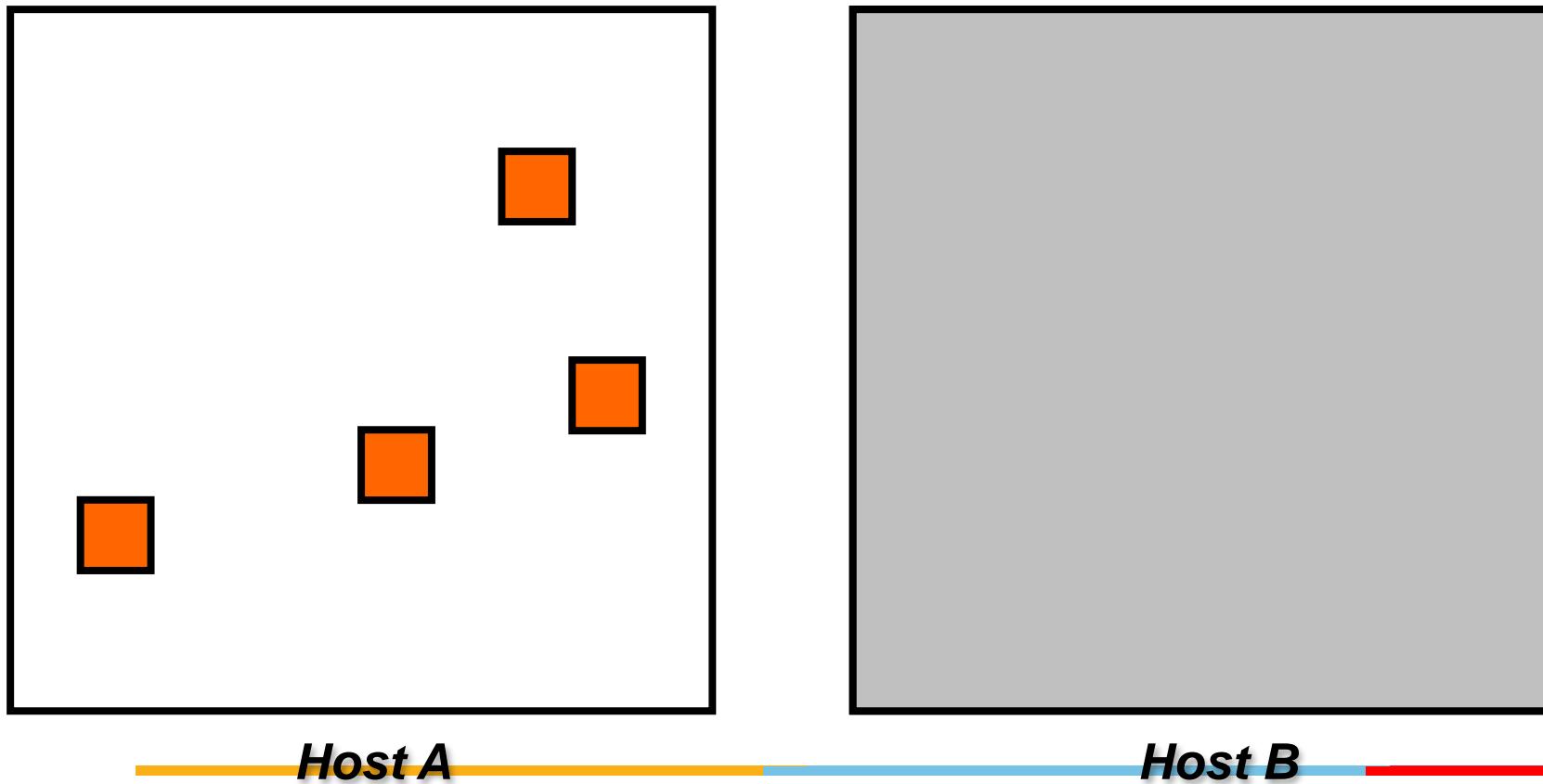
Pre-copy migration : Round 2



Live Migration Technique

Live migration process :

Stop and copy : Final Round



Live Storage Migration of Virtual Machine.

This migration technique constitutes moving the virtual disks or configuration file of a running virtual machine to a new data store without any interruption in the availability of the virtual machine's service.

References: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831656\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831656(v=ws.11))

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5071905> (A Live Storage Migration Mechanism over WAN for Relocatable Virtual Machine Services on Clouds)

Cloud Computing



BITS Pilani

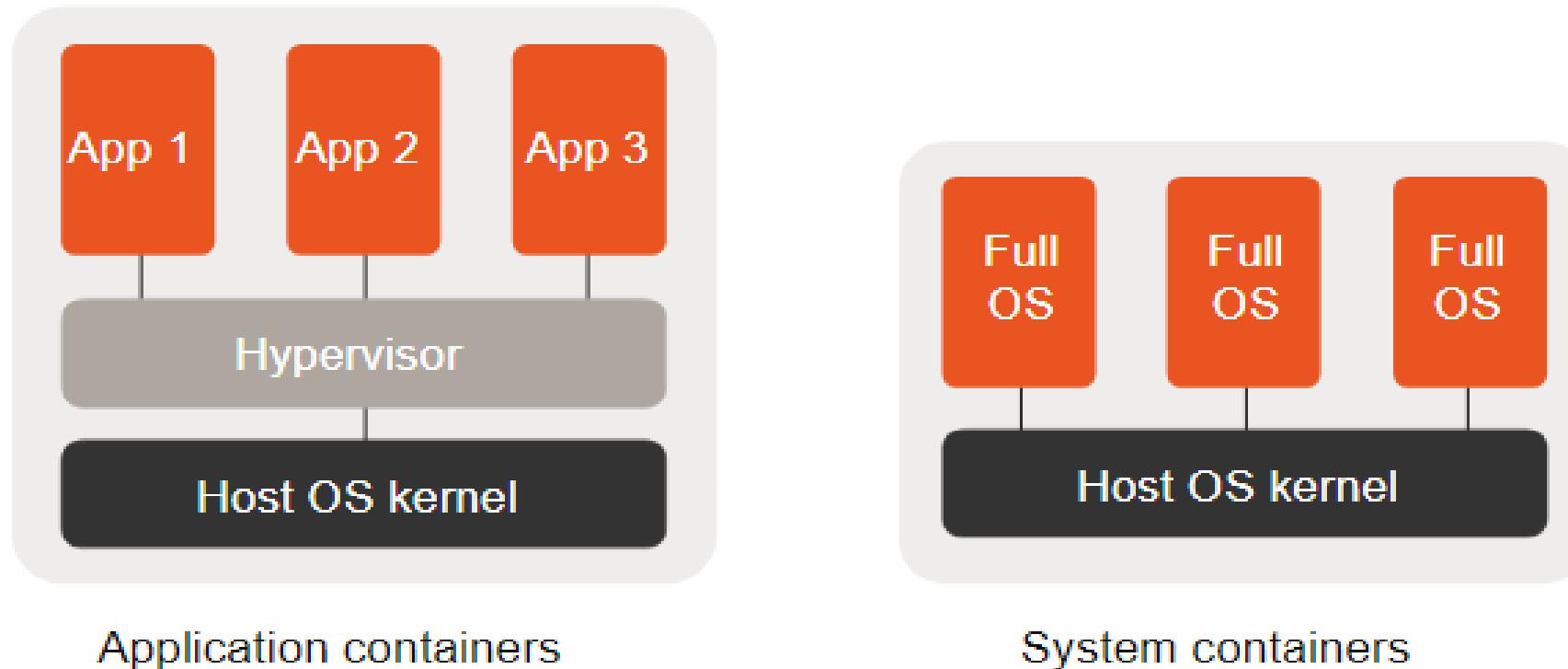
Module 4

(Linux containers, Dockers and Orchestration tools)

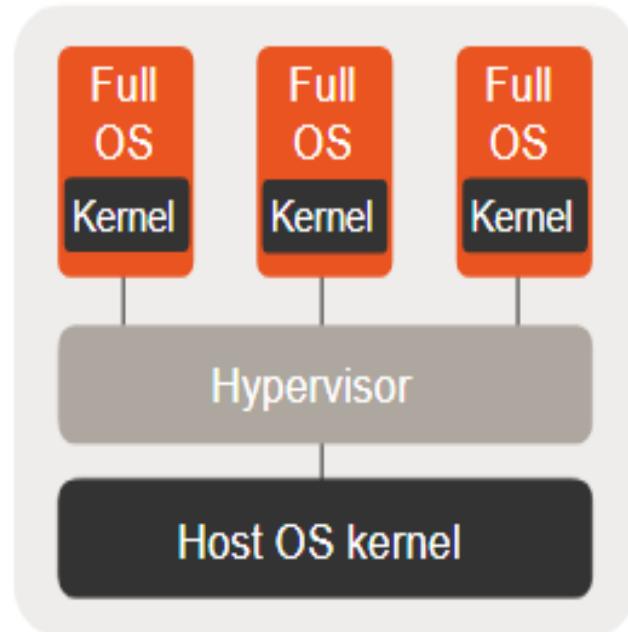
Agenda

- Linux Containers - LXC and LXD.
- Cloud orchestration technologies
- Dockers

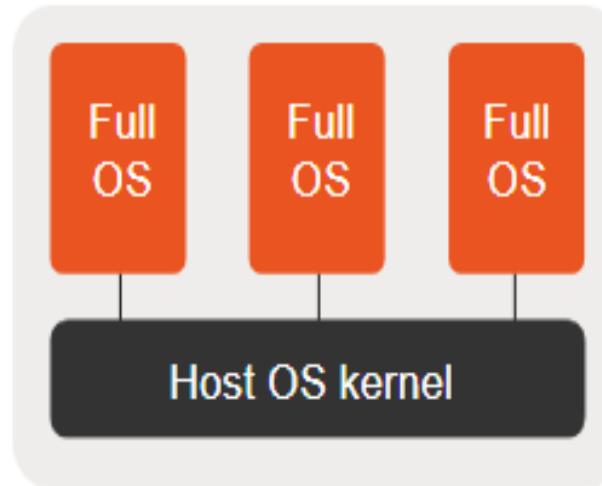
Difference Between Application Containers and System Containers



Difference Between System Containers and Virtual Machines



Virtual machines



System containers

LXC (Linux Containers)

- It is a system container.
- It is a set of 1 or more processes that are isolated from the rest of the system.
- It is used for Linux Containers which is an operating system that is used for running multiple Linux systems virtually on a controlled host via a single Linux kernel.
- It lets Linux users easily create and manage system or application containers.

Features provided by LXC :

- It provides Kernel namespaces such as IPC, mount, PID, network, and user.
- Apparmor and SELinux profiles
- Control groups (Cgroups).
- Seccomp policies
- Chroots (using pivot_root)

Goal: To create an environment as close as possible to a standard Linux installation but without the need for a separate kernel.

LXD (Linux container hypervisor)

- LXD is a next generation system container manager. It offers a user experience similar to virtual machines but using Linux containers instead.
- It is an extension of LXC with more functionality.
- It provides flexibility and scalability for various use cases, with support for different storage backends and network types and the option to install on hardware ranging from an individual laptop or cloud instance to a full server rack.
- It provides support for system containers and virtual machines.

Some Features of LXD are:

- Secure by design (unprivileged containers, resource restrictions and much more)
- Scalable (from containers on your laptop to thousand of compute nodes)
- Intuitive (simple, clear API and crisp command line experience)
- Image based (with a wide variety of Linux distributions published daily)
- Support for Cross-host container and image transfer (including live migration)
- Advanced resource control (cpu, memory, network I/O, block I/O, disk usage and kernel resources)
- Device passthrough (USB, GPU, unix character and block devices, NICs, disks and paths)

Installation and Launching Containers

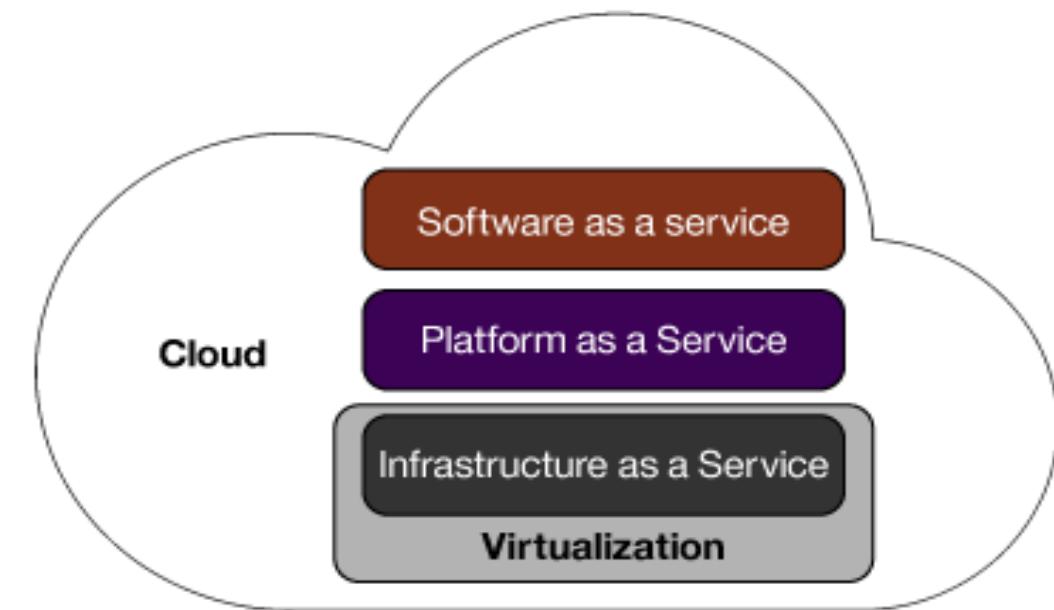
<https://www.cyberciti.biz/faq/install-lxd-on-ubuntu-20-04-lts-using-apt/>

1. lxc launch images:ubuntu/20.04 first
2. lxc info first
3. lxc stop first
4. lxc delete first
5. Lxc launch images:ubuntu/20.04 firstlimited –c limits.cp
u=2 –limits.memory=200MiB
6. lxc execute firstlimited –bash
7. cat /etc/*release

Virtualization and cloud computing

The manual approach to setting up environment included steps like these:

- Wait for approval
- Buy the hardware
- Install the OS
- Connect to and configure the network
- Get an IP
- Allocate the storage
- Configure the security
- Deploy the database
- Connect to a back-end system
- Deploy the application on the server



What challenges we face with the manual approach.?

Sol: backup, monitoring, networking, and configuring.

Significance of Configuration Management

The problem Context:

Suppose we have to deploy a software on top of hundreds of systems.

This software can be an operating system or a code or it can be an update of an existing software.

You can do this task manually, but what happens if you have to finish this task overnight because tomorrow might be a **Big Billion Day** sale in the company or some **Mega Sale** etc. in which heavy traffic is expected.

Even if you were able to do this manually there is a high possibility of multiple errors on your big day.

What if the software you updated on hundreds of systems is not working, then how will you revert back to the previous stable version, will you be able to do this task manually.

To solve the above issues:

Configuration Management was introduced. Chef, Puppet, etc. are configuration management tools that can automate the automate above tasks.

How it automates the above tasks?

We need to specify the configurations once on the central server and replicate that on thousands of nodes.

Configuration Management

It helps in performing the below tasks in a very structured and easy way:

- Figuring out which components to change when requirements change.
- Redoing an implementation because the requirements have changed since the last implementation.
- Reverting to a previous version of the component if you have replaced with a new but flawed version.
- Replacing the wrong component because you couldn't accurately determine which component was supposed to be replaced.

Cloud Orchestration Technologies

Why Orchestration technologies are required?

- To quickly configure, provision, deploy, and develop environments, integrate service management, monitoring, backup, and security services. All these services are repeatable.

They are used to address key challenges IaaS providers face when building a cloud infrastructure: **managing physical and virtual resources, namely servers, storage, and networks, in a holistic fashion.**

The orchestration of resources must be performed in a way to rapidly and dynamically provision resources to Applications. The software toolkit responsible for this orchestration is called a **virtual infrastructure manager (VIM)**.

Cloud Orchestration

What is Cloud orchestration?

- It is the end-to-end automation of the deployment of services in a cloud environment.
 - Manage cloud infrastructure: supplies and assigns required cloud resources to the customer like the **creation of VMs, allocation of storage capacity, management of network resources, and granting access to cloud software.**
- Cloud Orchestration automates provisioning of multiple servers, storage, databases and networks to make deployment and management of processes and resources smoother.
- It also ensures the complete maintenance of cloud elements in an integrated and harmonized way.

Objective: To accelerate the delivery of IT services while reducing costs.

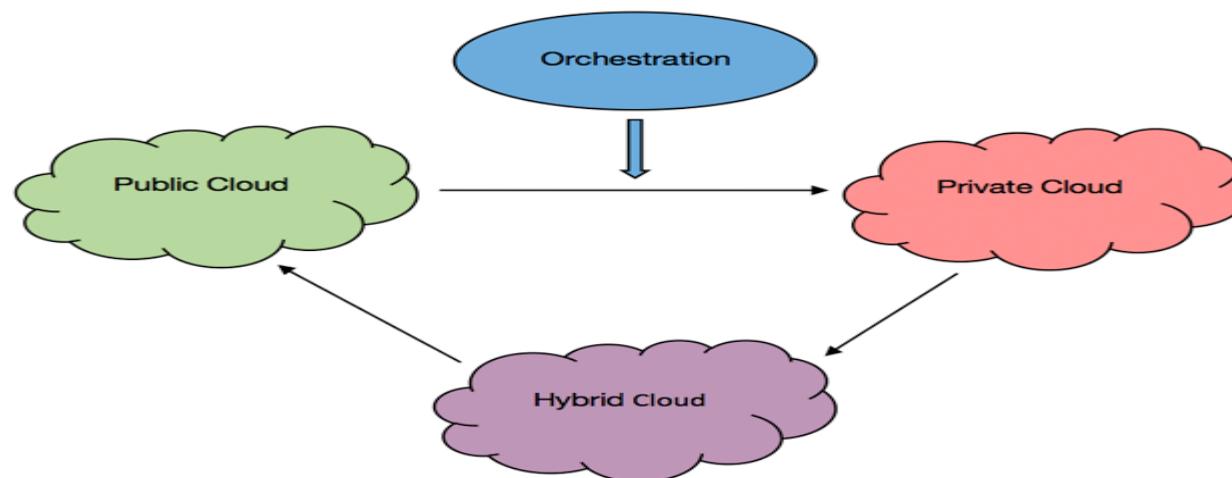
Benefits of Cloud Orchestration

- Effective Visibility and Control through dashboard
- Cost-Effective
- Reduced Errors

Three aspects of cloud orchestration:

- **Resource orchestration**, where resources are allocated
- **Workload orchestration**, where workloads are shared between the resources
- **Service orchestration**, where services are deployed on servers or cloud environments

The orchestration automates the services in all types of clouds—public, private, and hybrid.



Difference between orchestration and automation

- Automation usually focuses on a single task, **while orchestration deals with the end-to-end process, including management of all related services, taking care of high availability (HA), post deployment, failure recovery, scaling, and more.**
- A single task is involved in cloud automation whereas in cloud orchestration It is concerned with combining multiple such tasks into workflows.
- **Some examples of cloud automation** are Launching a web server, configuring a web server and some cases of cloud orchestration are Combining automated tasks like **launching web server and configuring a web service into a single workflow to meet client requests**

Note:

Note: Orchestrating is a process of automating a series of individual tasks to work together.

Orchestration tools

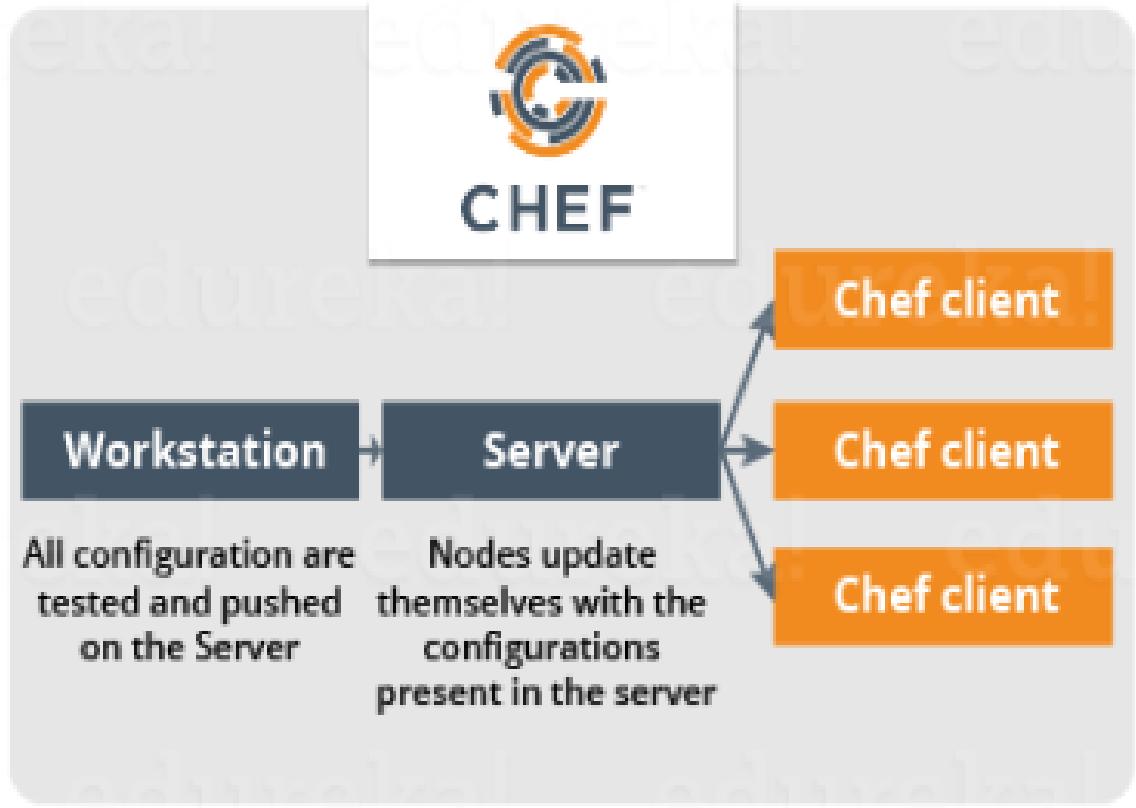
Cloud orchestration has useful tools that can automatically monitor cloud resources and modify VM instances while reducing manual work and work hours and they ensure efficient control on the cloud ecosystem.

1. Chef
2. Puppet
3. OpenStack (Already discussed)
4. Heat
5. Juju
6. Docker (Docker Swarm)

1. Chef

- It is used in infrastructure automation and allow us to automate and control vast numbers of computers in an automated, reliable, and scalable manner.
- Chef translates system administration tasks into reusable definitions, known as cookbooks and recipes.
- Cookbooks are the configuration units that allow us to configure and perform specific tasks within Chef on our remote nodes
- In a recipe, Chef authors define a system's desired state by writing configuration code. It specifies which resources to use and the order in which they are to be used.
- Chef then processes that code along with data about the specific node where the code is running to ensure that the desired state actually matches the state of the system.
- Chef uses a pure-Ruby, domain-specific language (DSL) for writing system configurations. It is developed on the basis of Ruby DSL language. It is used to streamline the task of configuration and managing the company's server.

Chef Architecture



Ref: <https://www.edureka.co/blog/chef-tutorial/>

Functionality:

Chef allows us to dynamically provision and de-provision your infrastructure on demand to keep up with peaks in usage and traffic.

- It enables new services and features to be deployed and updated more frequently, with little risk of downtime.
- With Chef, we can take advantage of all the flexibility and cost savings that cloud offers.

Note: The **Chef client** is an agent that runs on a node and performs the actual tasks that configure it.

- Chef can manage anything that can run the Chef client, like physical machines, virtual machines, containers, or cloud-based instances.

Chef Architecture Cont...



CHEF WORKSTATION™

Chef Server

Clients

Chef Architecture Cont...

- **Pull Configuration:** Nodes will automatically update themselves with the configurations present in the Server.
 - Chef supports multiple platforms like AIX, RHEL/CentOS, FreeBSD, OS X, Solaris, Microsoft Windows and Ubuntu.
 - Chef can be integrated with cloud-based platforms such as Amazon EC2, Google Cloud Platform, OpenStack, SoftLayer, Microsoft Azure and Rackspace to automatically provision and configure new machines.

Example demonstration of creating a chef book:

<https://www.digitalocean.com/community/tutorials/how-to-create-simple-chef-cookbooks-to-manage-infrastructure-on-ubuntu>

Chef cookbooks

- Chef uses **cookbooks** to determine how each node should be configured.
- Cookbooks are usually used to handle one specific service, application, or functionality. For instance, **a cook book can be created to set and sync the node's time with a specific server. It may install and configure a database application. Cookbooks are basically packages for infrastructure choices.**
- Cookbooks consist of multiple **recipes**; a recipe is an automation script for a particular service that's written using the Ruby language.
- Cookbooks are created on the workstation and then uploaded to a Chef server.
- From there, recipes and policies described within the cookbook can be assigned to nodes as part of the node's "run-list".
- A run-list is a sequential list of recipes and roles that are run on a node by chef-client in order to bring the node into compliance with the policy you set for it.

Note1: <https://www.linode.com/docs/guides/beginners-guide-chef/> and https://docs.chef.io/chef_overview/

Note2: <https://www.digitalocean.com/community/tutorials/how-to-create-simple-chef-cookbooks-to-manage-infrastructure-on-ubuntu>

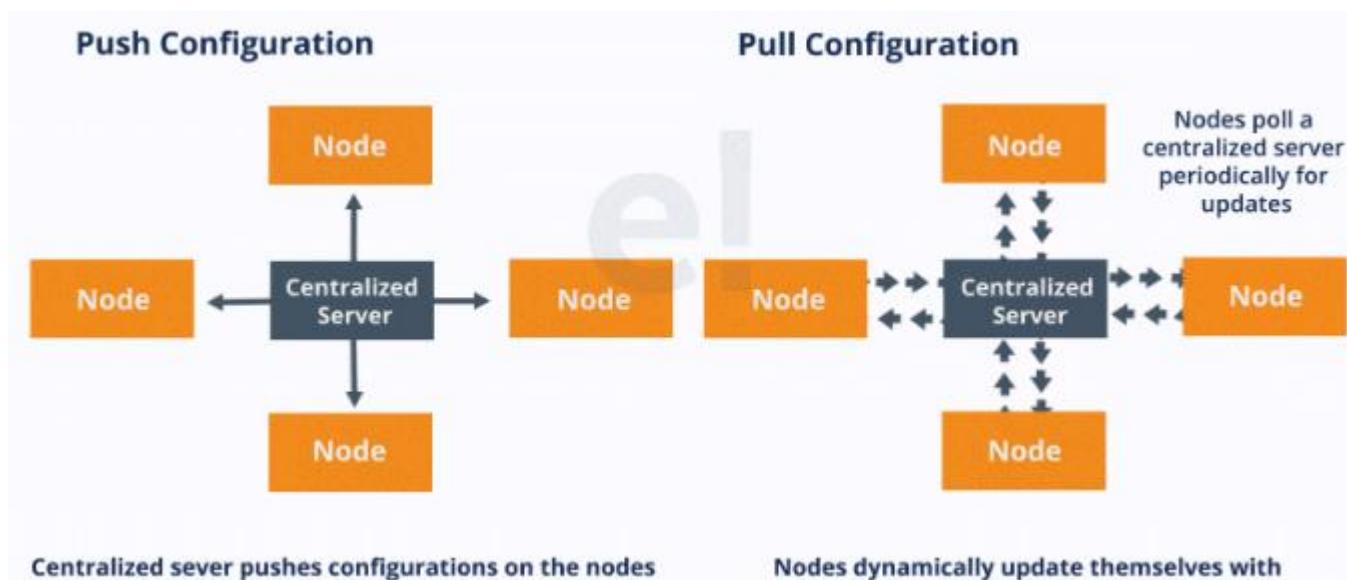
There are broadly two ways to manage your configurations namely Push and Pull configurations.

Pull Configuration: In this type of Configuration Management, the nodes poll a centralized server periodically for updates. These nodes are dynamically configured so basically they are pulling configurations from the centralized server.

Pull configuration is used by tools like Chef, Puppet etc.

Push Configuration: In this type of Configuration Management, the centralized Server pushes the configurations to the nodes. Unlike Pull Configuration, there are **certain commands that have to be executed in the centralized server** in order to configure the nodes.

Push configuration is used by tools like Ansible and Salt Stack



Ref: <https://www.edureka.co/blog/what-is-chef/>

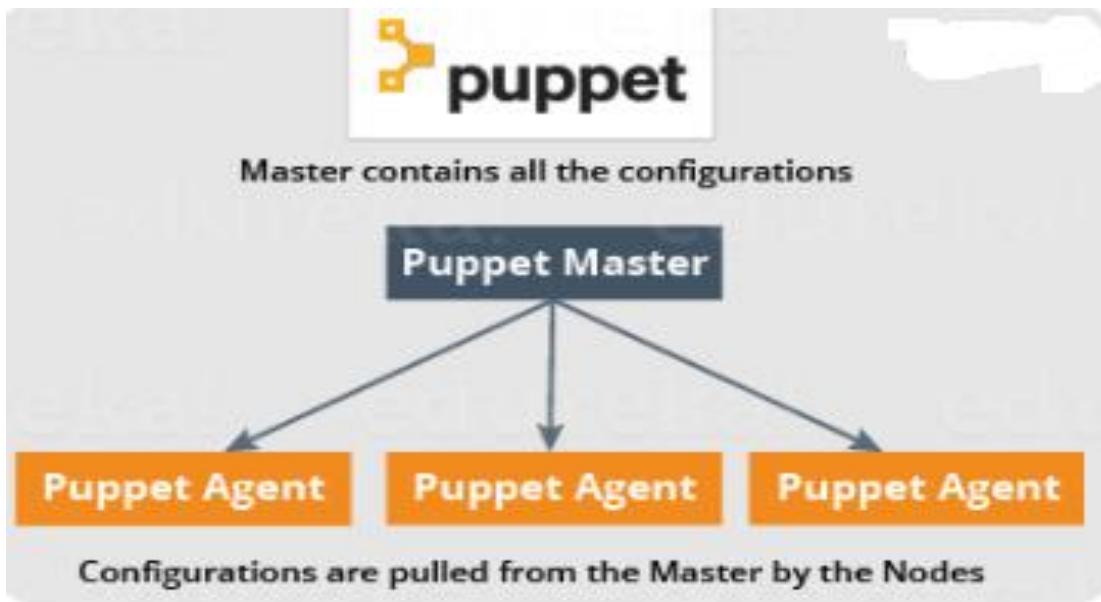
2. Puppet

What is Puppet?

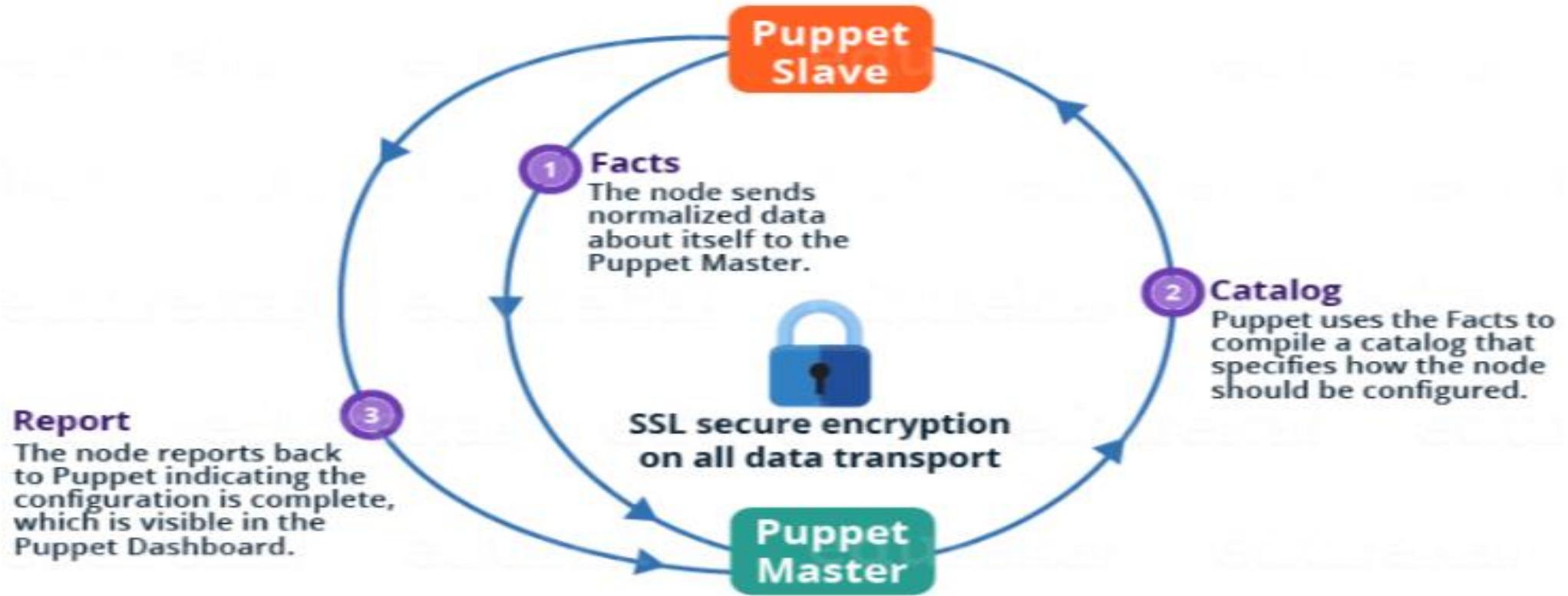
- Puppet is a Configuration Management tool and can be used to install and manage software on existing server instances (e.g., installation of packages, starting of services, installing scripts or config files on the instance).
- Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across the whole infrastructure of an organization.
- Puppet is specially designed to manage the configuration of Linux and Windows systems.
- It is written in Ruby and uses its unique **Domain Specific Language (DSL)** to describe system configuration.
- They do the heavy lifting of making one or many instances perform their roles without the user needing to specify the exact commands. No more manual configuration or ad-hoc scripts are needed.

Puppet (Cont...)

- It requires installation of a master server and client agent in target nodes, **and includes an option for a standalone client.**



Puppet



<https://www.edureka.co/blog/puppet-tutorial/>

Puppet Versions

Puppet comes in two versions:

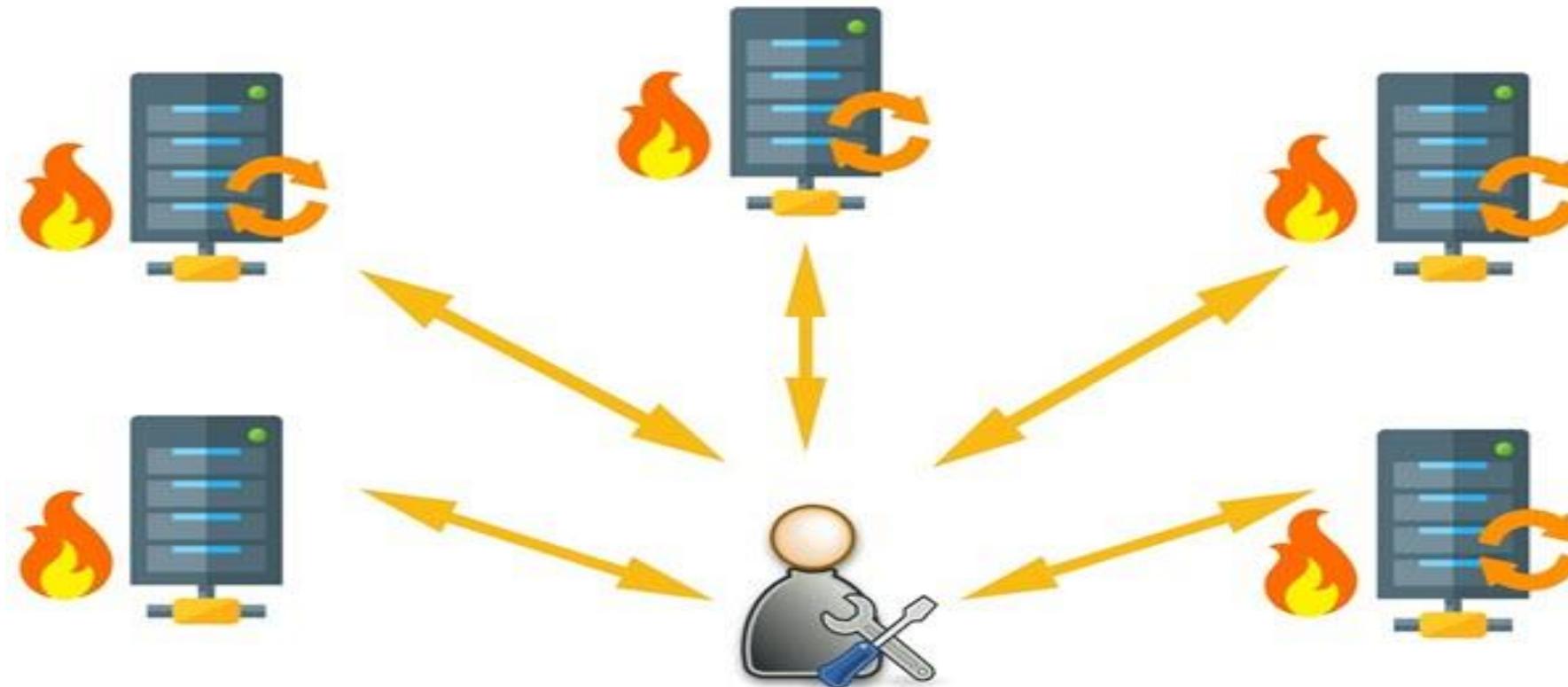
Open Source Puppet: It is a basic version of Puppet configuration management tool, which is also known as Open Source Puppet. It is available directly from Puppet's website and is licensed under the Apache 2.0 system.

Puppet Enterprise: Commercial version that offers features like compliance reporting, orchestration, role-based access control, GUI, API and command line tools for effective management of nodes.

Puppet Functionalities

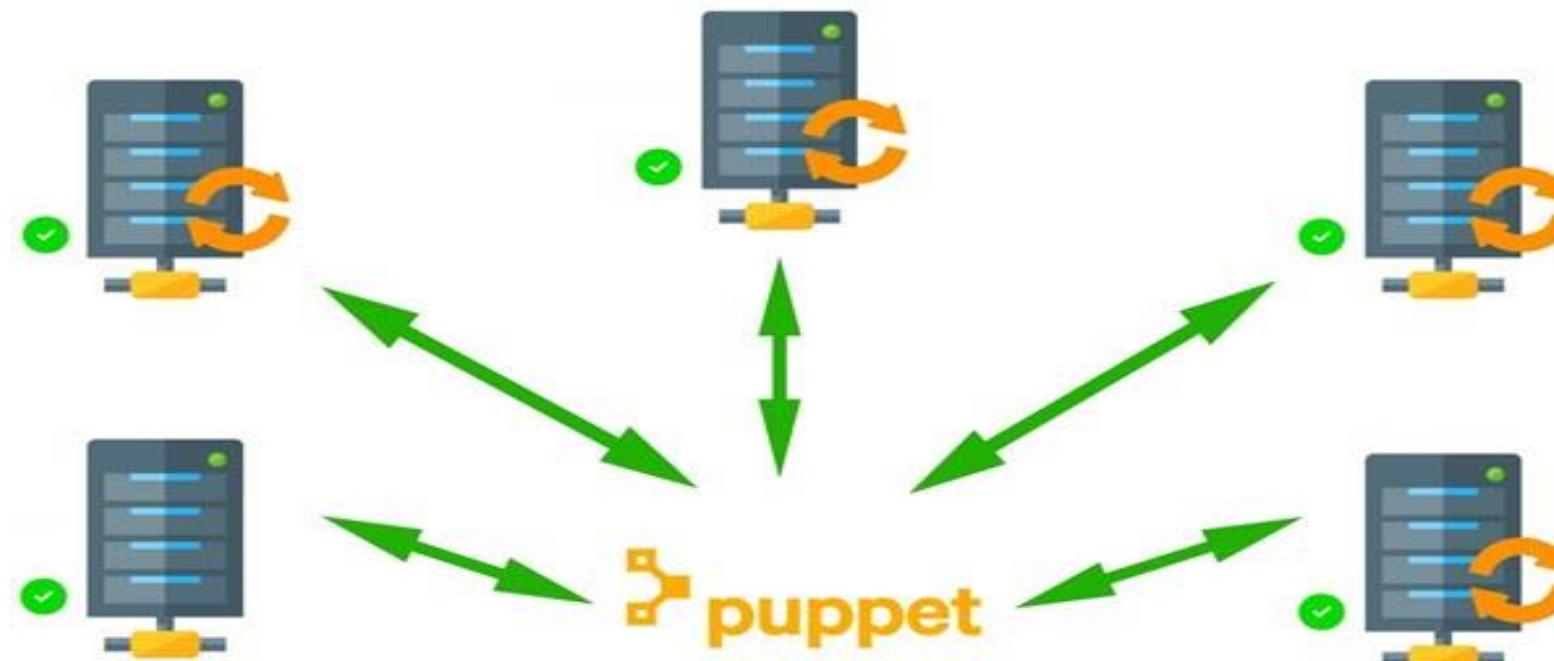
- **Defining distinct configurations** for each and every host, and continuously checking and confirming whether the required configuration is in place or not.
- Dynamic **scaling-up and scaling-down** of machines.
- Providing **control over all your configured machines**, so a centralized (master-server) change gets propagated to all, automatically.

What Puppet Can do?



Note: The role of **system admin** is to ensure that all these servers are always up to date and running with full functionality.

What Puppet Can do? (Cont...)



Note: Puppet here allows you to write a simple code which can be deployed automatically on these servers. This reduces the human effort and makes the development process fast and effective.

How Puppet Works?

- Puppet is based on a **Pull deployment model**, where the agent nodes check in regularly after every **1800** seconds with the master node to see if anything needs to be updated in the agent.
- If anything needs to be updated the agent pulls the necessary puppet codes from the master and performs required actions.

Example: Master - Agent Setup:

The Master:

A Linux based machine with Puppet master software installed on it. It is responsible for maintaining configurations in the form of puppet codes. The master node can only be Linux.

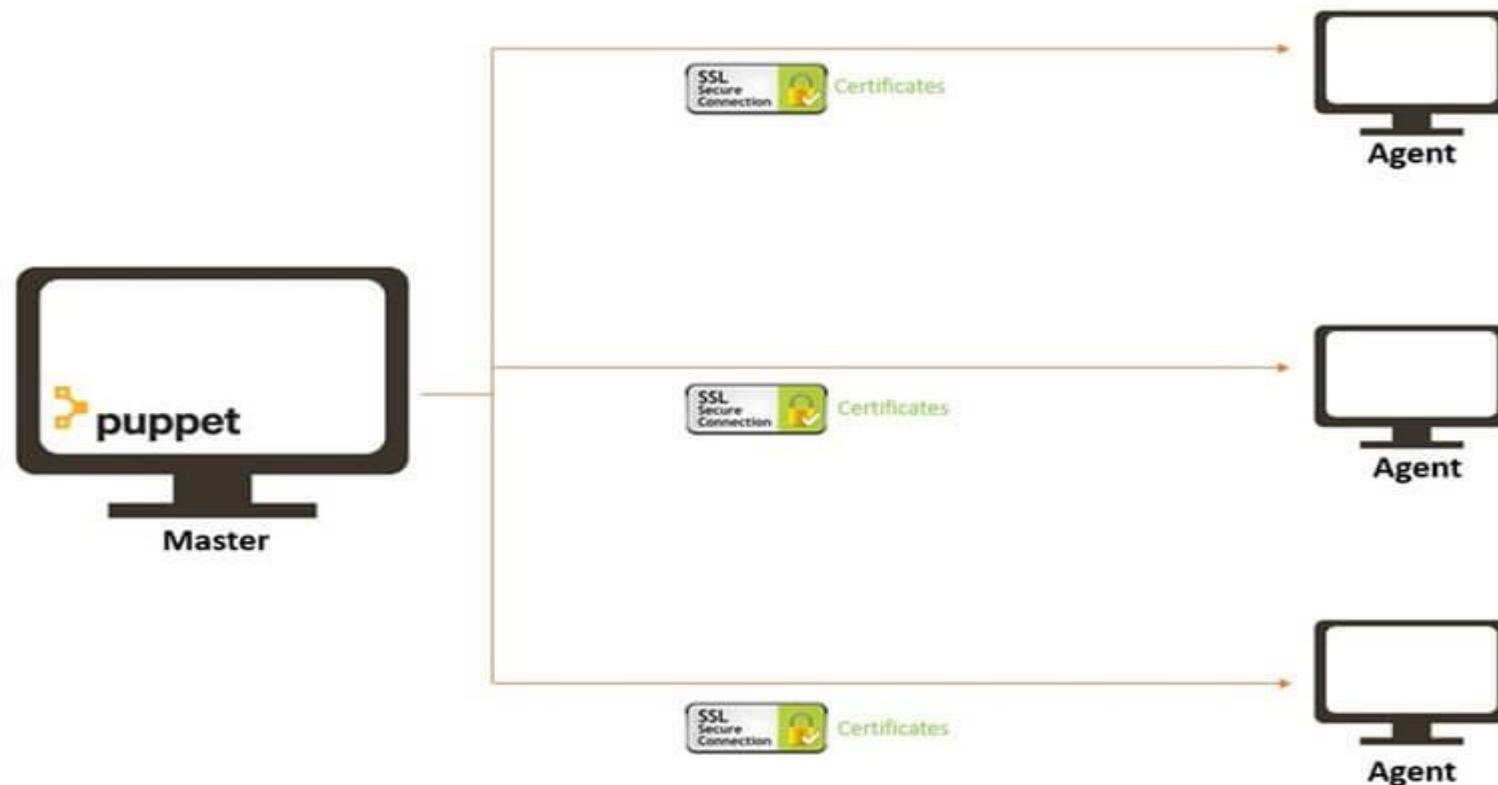
The Agents:

The target machines managed by a puppet with the puppet agent software installed on them.

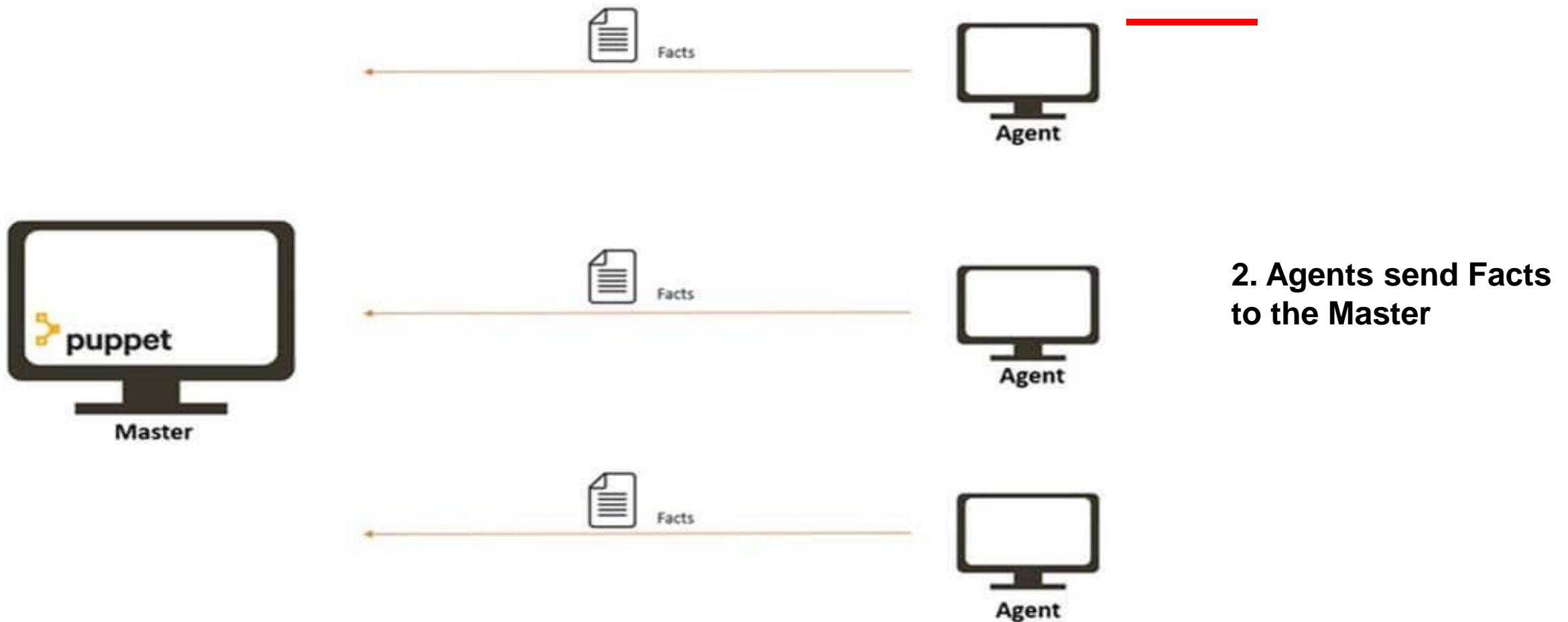
How Puppet Works? (Cont...)

- The agent can be configured on any supported operating system such as Linux or Windows or Solaris or Mac OS.
- The communication between master and agent is established through secure certificates.

1. Connection Establishment



How Puppet Works? (Cont...)



Facts contain information that includes the hostname, kernel details, IP address, file name details, etc.

How Puppet Works? (Cont...)



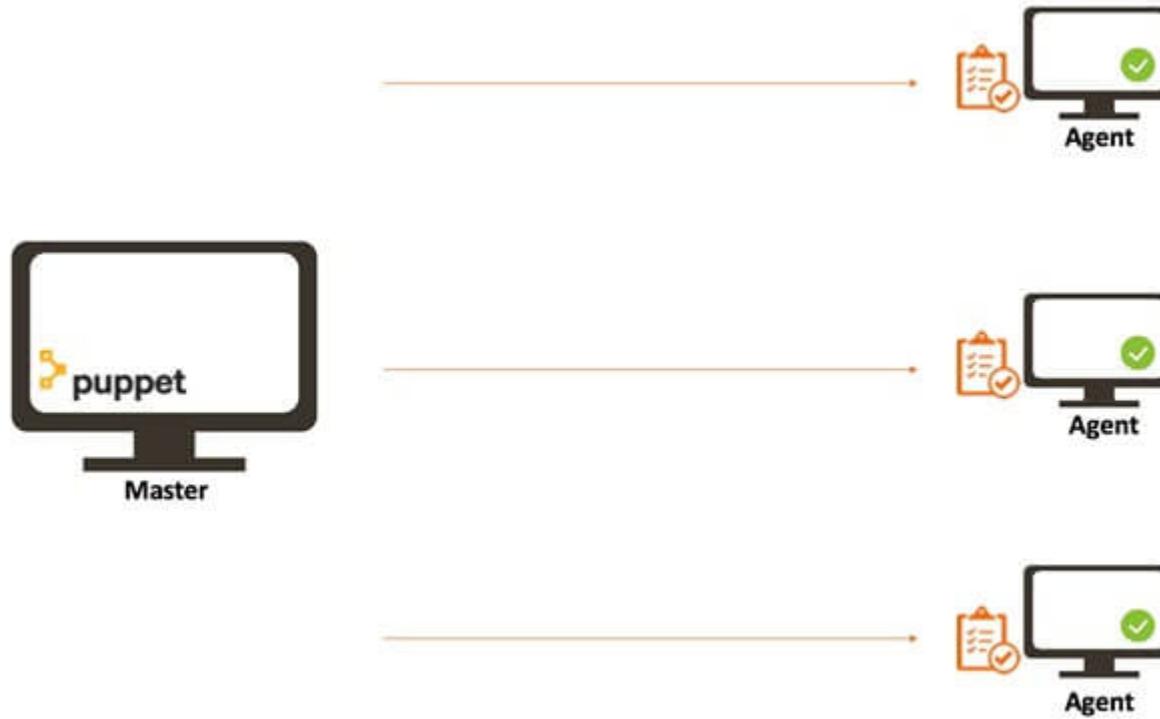
3. Master sends catalog to Agent



Master sends catalog to Agent

Puppet Master uses Facts' data and compiles a list with the configuration to be applied to the agent. This list of configuration to be performed on an agent is known as a **catalog** (package installation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.).

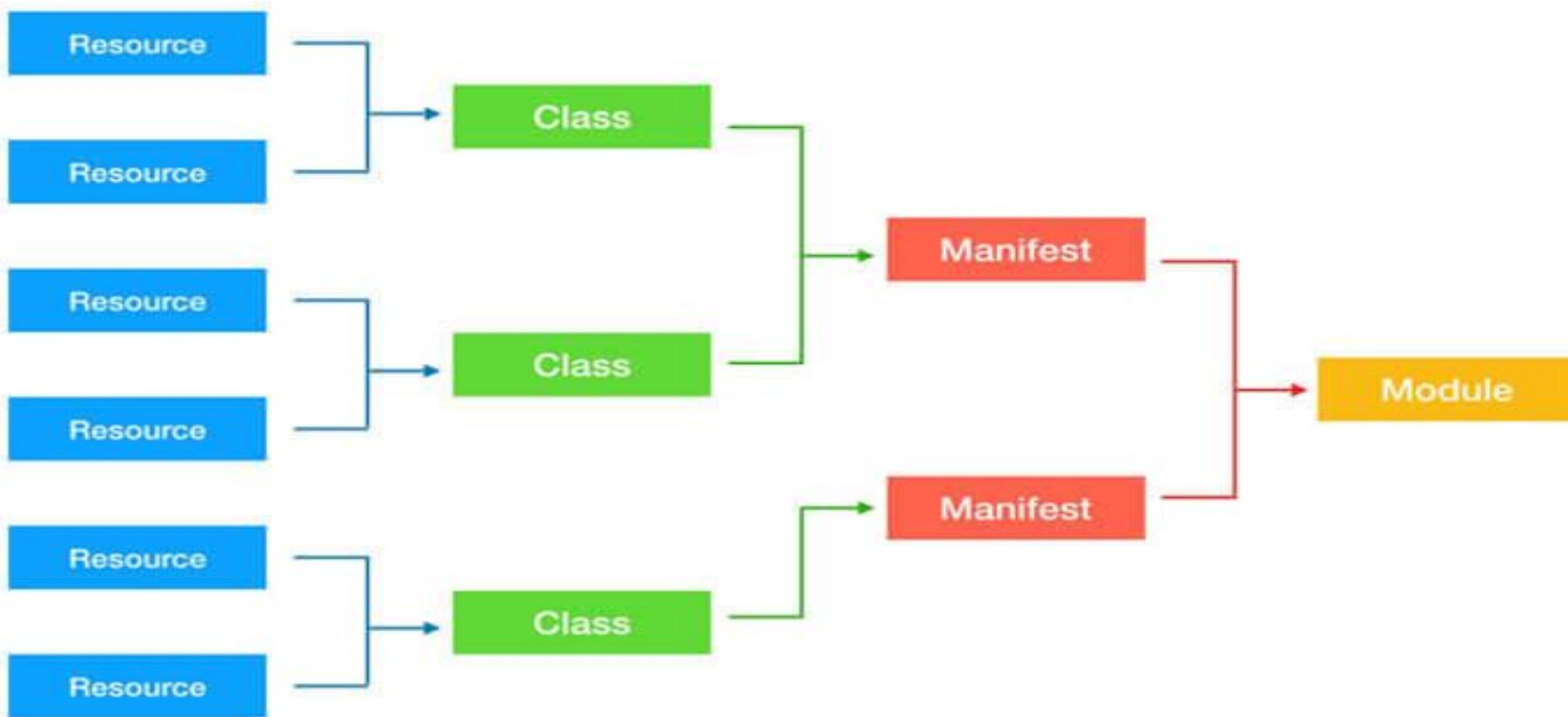
How Puppet Works? (Cont...)



4. Each Agents applies configuration

Note: Once the node apply configuration, then the node reports back to puppet master indicating that the configuration has been applied and completed.

Puppet Blocks



Puppet Blocks (Cont...)

Puppet Resources:

Puppet Resources are the building blocks of Puppet.

Resources are the **inbuilt functions** that run at the back end to perform the required operations in puppet.

Puppet Classes:

A combination of different resources can be grouped together into a single unit called class.

Puppet Manifest:

Manifest is a directory containing puppet DSL files. Those files have a .pp extension. The .pp extension stands for puppet program. The puppet code consists of definitions or declarations of Puppet Classes.

Puppet Modules:

Modules are a collection of files and directories such as Manifests, Class definitions. They are the re-usable and sharable units in Puppet.

Ref:<https://www.guru99.com/puppet-tutorial.html> and <https://www.guru99.com/devops-tutorial.html>

Differences between Puppet and Chef

How the two platforms (Puppet and Chef) stack up against one another:

- **Puppet is geared toward system admins** who need to specify configurations like dependencies, whereas **Chef is for developers** who actually write the code for the deployment.

Differences between Puppet and Chef (Cont...)

While both tools have similar goals, the means used to achieve them differ. Some of them are provided below:

2. How resources are described:

Puppet uses a **declarative language** that is similar to JSON or XML.

You describe the resource's state but cannot intervene in how this state is achieved.

Chef uses an **imperative language**. This means you more or less have full-featured Ruby at your disposal.

For example, **Puppet is like writing configuration files** whereas using Chef is like **programming the control of your nodes**.

If you or your team have more experience with system administration, you may prefer Puppet. On the other hand, if most of you are developers, Chef might be a better fit.

Differences between Puppet and Chef (Cont...)

2. Configuration files:

In Puppet, you create **manifests and modules**, while in Chef you deal with **recipes and cookbooks**. Manifests and recipes usually describe **single resources** while **modules and cookbooks** describe the more general concepts (a LAMP server running your application, for instance).

Examples of resources with which one can deal with are files, directories, network interfaces, and applications. Commands like mkdir, cat, and apt-get or yum are replaced with desired states (“present,” “absent,” “updated,” and so on).

Differences between Puppet and Chef (Cont...)

Examples of resource definitions for a **directory** and a **file**

Example #1 — Creating a Directory

```
# Puppet
file { '/tmp/example'
  ensure => 'directory',
}

# Chef
directory '/tmp/example'
```

Example #2 — Writing Content to a File

```
# Puppet
file { '/tmp/hello':
  ensure => 'present',
  content => "hello, world!",
}

# Chef
file '/tmp/hello' do
  content 'hello, world!'
end
```

Differences between Puppet and Chef (Cont...)

Example #3 — Installing and Enabling Apache

```
# Puppet

class apache2 {
    # Package name differs between distributions.

    # Here we select appropriate one

    if $::osfamily == 'RedHat' {
        $apachename = 'httpd'
    } elseif $::osfamily == 'Debian' {
        $apachename = 'apache2'
    } else {
        print "This is not a supported distro."
    }
}

package { 'apache':
    name => $apachename,
    ensure => 'present',
}

service { 'apache-service':
    name => $apachename,
    enable => true,
    ensure => 'running',
}
```

Differences between Puppet and Chef (Cont...)

Examples of resource definitions for a installing and Enabling Apache using Chef

```
# Chef

case node[:platform]
when 'ubuntu', 'debian'
  # Update apt cache every 86400 seconds (that is once a day)
  apt_update 'Update the apt cache daily' do
    frequency 86_400
    action :periodic
  end

  apachename = 'apache2'
when 'redhat', 'centos'
  apachename = 'httpd'
end

# Install 'apache2' package
package 'Install apache2' do
  package_name apachename
end

# Enable and start the service
service apachename do
  supports :status => true
  action [:enable, :start]
end
```

3. OpenStack

Note: It is already discussed and please refer to module 3 slides that was shared with you on CMS

4. Heat

- **Heat** is the main project in the **OpenStack Orchestration program**.
- **Heat** allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application.
- In many ways, Heat does for applications what OpenStack infrastructure components (Nova, Cinder, and the like) do for vendor hardware and software—it simplifies the integration.

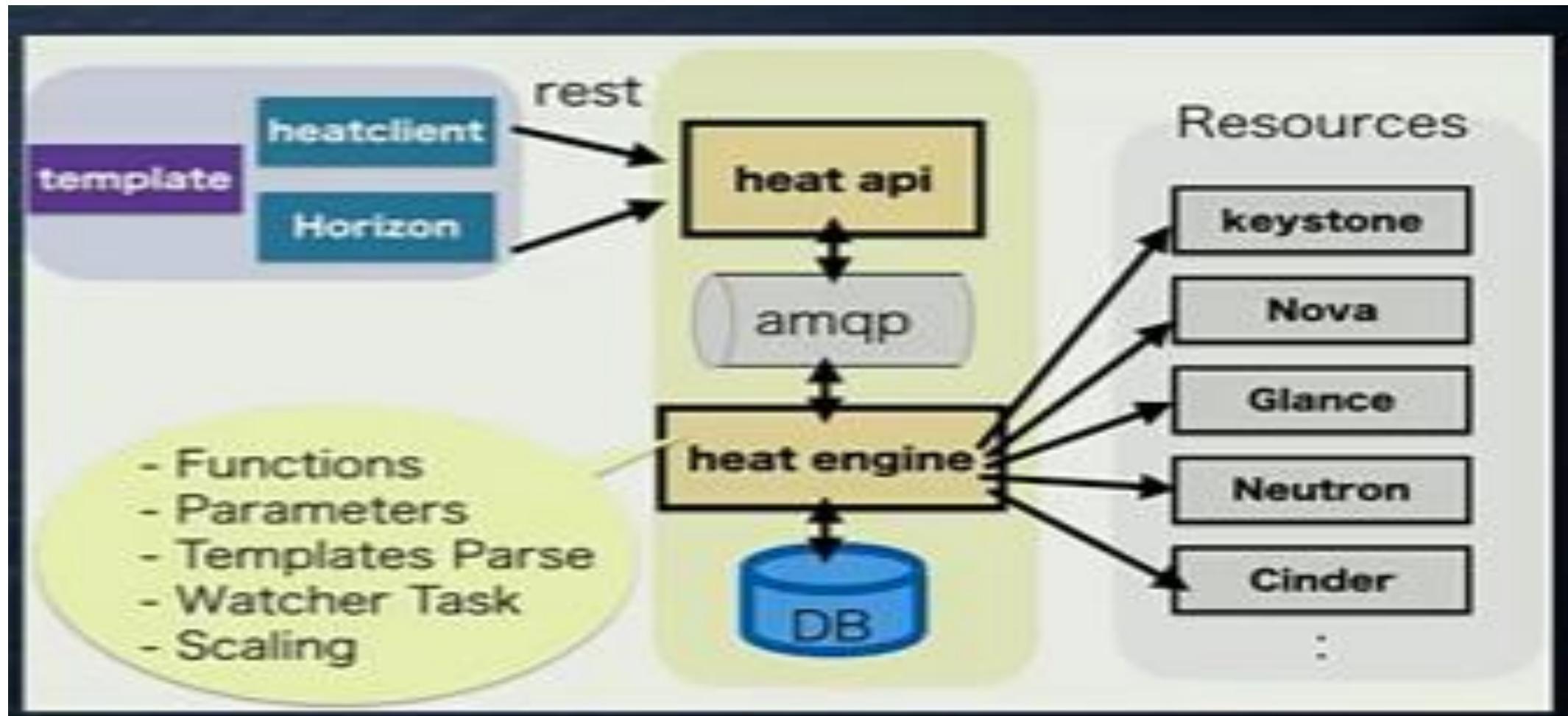
4. Heat (Cont...)

- It enables orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code.
- A native Heat template format is evolving, but Heat also endeavours to provide compatibility with the [AWS CloudFormation](#) template format, so that many existing CloudFormation templates can be launched on OpenStack.

heat-engine:

The heat engine does the main work of orchestrating the launch of templates and providing events back to the API consumer.

How heat works?



How heat works? (Cont....)

- Heat manages the whole lifecycle of the application - when you need to change your infrastructure, simply modify the template and use it to update your existing stack.
- A Heat template describes the infrastructure for a cloud application in a text file that is readable and writable by humans, and can be checked into version control, diffed, etc.
- Infrastructure resources that can be described include: servers, floating ips, volumes, security groups, users, etc.
- Heat knows how to make the necessary changes. It will delete all of the resources when you are finished with the application, too.
- Heat primarily manages infrastructure, but the templates integrate well with software configuration management tools such as **Puppet** and **Chef**.
- Heat also provides an autoscaling service that integrates with Telemetry, so you can include a scaling group as a resource in a template.
- Templates can also specify the relationships between resources (e.g. this volume is connected to this server).
- This enables Heat to call out to the OpenStack APIs to create all of your infrastructure in the correct order to completely launch your application.

Heat Orchestration Templates (HOT)

(HOT) are native to Heat and are expressed in YAML. These templates consist of:

- **Resources** (mandatory fields) are the OpenStack objects that you need to create, like server, volume, object storage, and network resources. These fields are required in HOT templates.
- **Parameters** (optional) denote the properties of the resources. Declaring the parameters can be more convenient than hard coding the values.
- **Output** (optional) denotes the output created after running the Heat template, such as the IP address of the server.

Each resource, in turn, consists of:

References—used to create nested stacks

Properties—input values for the resource

Attributes—output values for the resource

Basic HOT template

```
#template structure
heap_template_version:2015-04-30
#validate version of HOT
description:
# Template description (features available or functions)
parameters:
#Template input parameters (Security groups, which one to use, which SSH key to use, etc.)
resources:
#Resources to be created (e.g., virtual machines).
outputs:
#The output (IP address to login the created VM)
```

Creating and deploying a Heat Template

A template which will create the following and connect them up:

- A network
- A static IP
- A router
- An instance
- A floating IP
- A volume

The Heat template: The template must be in yaml format and be saved with the .yaml file extension.

Note: We should also make sure that you use the correct indentation.

Steps for deploying:

- Define a Heat template's required parameters in a YAML file, and
- Deploy the template via the command line (**OpenStack CLI**).

basic-stack.yaml

(Note: A Stack means a group of connected cloud resources)

```
heat_template_version: 2016-10-14

parameters:
  flavor:
    type: string
    description: I am using the smallest flavor available because i'll be spinning up a cirros instance. You can use an environment file to override the defaults.
    default: t1.tiny
    constraints:
      - custom_constraint: nova.flavor
  image:
    type: string
    description: This uses a cirros image but you can create an environment file to change the default values.
    default: cirros
    constraints:
      - custom_constraint: glance.image
  heat_volume_size:
    type: number
    label: Volume Size (GB)
    description: External Volume Size in GB
    default: 1
```

```
resources:
  heat_volume:
    type: OS::Cinder::Volume
    properties:
      size: { get_param: heat_volume_size }
  heat_volume_attachment:
    type: OS::Cinder::VolumeAttachment
    properties:
      volume_id: { get_resource: heat_volume }
      instance_uuid: { get_resource: heat_server }
  heat_network:
    type: OS::Neutron::Net
    properties:
      admin_state_up: true
      name: heat_network
  heat_network_subnet:
    type: OS::Neutron::Subnet
    properties:
      network: { get_resource: heat_network }
      cidr: "10.1.1.0/24"
      dns_nameservers: ["8.8.8.8"]
      gateway_ip: "10.1.1.1"
      ip_version: 4
```

```
heat_router:
  type: OS::Neutron::Router
  properties:
    external_gateway_info: { network: internet }
    name: heat_router
heat_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router_id: { get_resource: heat_router }
    subnet: { get_resource: heat_network_subnet }
heat_server_port:
  type: OS::Neutron::Port
  properties:
    network: { get_resource: heat_network }
    fixed_ips:
      - subnet_id: { get_resource: heat_network_subnet }
heat_server:
  type: OS::Nova::Server
  properties:
    name: heat_server
    flavor: { get_param: flavor }
    image: { get_param: image }
```

```
networks:
  - port: { get_resource: heat_server_port}
heat_server_public_ip:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network: "internet"
heat_server_ip_assoc:
  type: OS::Neutron::FloatingIPAssociation
  properties:
    floatingip_id: { get_resource: heat_server_public_ip }
    port_id: { get_resource: heat_server_port }

outputs:
  heat_server_public_ip:
    description: IP Address of the deployed heat_server instance
    value: { get_attr: [ heat_server_public_ip, floating_ip_address ]}
```

Deploying the stack

Create a stack by running the following command in the OpenStack CLI:

openstack stack create -t <template name> <stack name>

For example, for the template above, run:

```
$ openstack stack create -t basic-stack.yaml basic-stack
```

This command will return the following:

Field	Value
id	6be269ec-d22f-4cf0-bf04-df71cc6dcf75
stack_name	basic-stack
description	No description
creation_time	2019-02-12T15:58:20Z
updated_time	None
stack_status	CREATE_IN_PROGRESS
stack_status_reason	Stack CREATE started

References of Heat

<https://docs.ukcloud.com/articles/openstack/ostack-how-create-heat-template.html>

<https://clouddocs.f5.com/cloud/openstack/v1/heat/how-to-deploy-heat-stack.html>

Ref:<https://wiki.openstack.org/wiki/Heat>

<https://developer.ibm.com/articles/cl-cloud-orchestration-technologies-trs/>

<https://livebook.manning.com/book/openstack-in-action/chapter-12/11>

5. Juju

What is Juju?

- Juju is a state-of-the-art, open source modelling tool for operating software in the cloud.
- Juju allows us to deploy, configure, manage, maintain, and scale cloud applications quickly and efficiently on public clouds as well as physical servers, OpenStack, and containers.
- We can use Juju from the command line or through its beautiful GUI.
- The goal of Juju is to reuse of common operational code in widely different environments.
- It is an open source automatic service orchestration management tool developed by Canonical, the developers of the Ubuntu OS.

Application modelling?

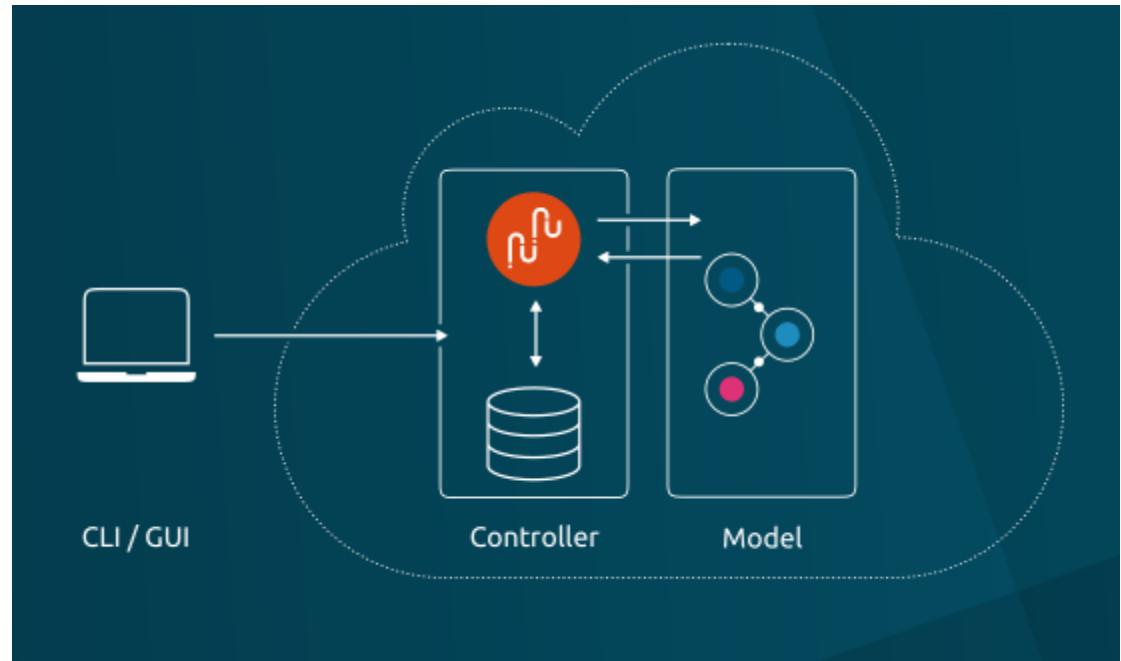
- In modern environments, Even simple applications may require several other applications in order to function - like a database and a web server.
- For modeling a more complex system, e.g. OpenStack, many more applications need to be installed, configured and connected to each other.
- **Juju's application modelling provides tools to express the intent of how to deploy such applications and to subsequently scale and manage them.**
- With Juju, you create a model of the relationships between applications that make up our solution and we have a mapping of the parts of that model to machines. Juju then applies the necessary configuration management scripts to each machine in the model.
- Application-specific knowledge such as dependencies, scale-out practices, operational events like backups and updates, and integration options with other pieces of software are encapsulated in Juju's 'charms'.

How Juju works?

Juju controller: the heart of Juju

Juju's controller manages all the machines in our running models, responding to the events that are triggered throughout the system.

It also manages scale out, configuration and placement of all your models and applications.



Terminologies of Juju

Charms

Charms are sets of scripts for deploying and operating software. With event handling built in, they can declare interfaces that fit charms for other services, so relationships can be formed.

Bundles

Bundles are collections of charms that link services together, so you can deploy whole chunks of app infrastructure in one go.

What they do:

Install

Configure

Connect

Upgrade and update

Scale out and scale back

Perform health checks

Undertake operational actions

charms

- Juju utilizes **charms**, which are open source tools that simplify specific deployment and management tasks.
- The charm defines everything you all collaboratively know about deploying that particular application brilliantly
- Charms encapsulate application configurations, define how services are deployed, how they connect to other services, and how they are scaled. Also define how services integrate, and how their service units react to events in the distributed environment, as orchestrated by Juju.
- After a service is deployed, Juju can define relationships between services and expose some services to the outside world.

Notes:

- A Juju charm usually includes all of the intelligence needed to scale a service horizontally by adding machines to the cluster, preserving relationships with all of the services that depend on that service.
- Juju provides both a command-line interface and an intuitive web application for **designing, building, configuring, deploying, and managing your infrastructure. Juju automates the mundane tasks, allowing you to focus on creating amazing applications.**
- **Charm store** includes a collection of charms that let you deploy whatever services you like in Juju.

Charms (Cont...)

What language are charms written in?

Charms can be written in any language or configuration management scripting system. Chef and Puppet are common, **more complex charms tend to use Python.**

Charms have been written in **Ruby**, **PHP**, and many charms are a collection of simple bash scripts.

What charms are currently available?

Charms are available for hundreds of common and popular cloud-oriented applications such as **MySQL**, **MongoDB**, and others, with new ones being added every day. Check out the public charm store for an up to the minute list of charms: [Juju Charm Store](#)

Application scenario with Charms and Bundles

- Modern applications are typically composed of many applications - databases, front-ends, big data stores, logging systems, key value stores.
- Each application will be defined by a **single charm**.
- To describe the whole application you need to describe the set of charms and their relationships - what is connected to what - and we use a **bundle** for that.

Example Application with Juju (Charms and Bundles)

- A **content-management system** bundle could specify a database and a content management server, together with key-value stores and front-end load-balancing systems.
- Each of those applications are described by a **charm**; the bundle describes **the set of charms**, their configuration, and the relationships between them.
- This allows development teams to share not only the core primitive for each application, but also enables sharing higher-level models of several applications.
- It allows you to replicate complex application models in a cloud just by dropping the same bundle onto your Juju GUI.
- Bundles can be shared publicly or privately.
- New bundles are added to the public collection every week. And you can now bootstrap Juju, launch the GUI and deploy a bundle with a single command. That means you can go from zero to a full cloud deployment in seconds with the right bundle.

Benefits of Juju

- Juju is the fastest way to model and deploy applications or solutions on all major public clouds and containers.
- It helps to reduce deployment time from days to minutes.
- Juju works with existing configuration management tools, and can scale workloads up or down very easily.
- No prior knowledge of the application stack is needed to deploy a Juju charm for the product.
- **Juju includes providers for all major public clouds, such as Amazon Web Services, Azure, and HP as well as OpenStack, MAAS, and LXC containers.**
- It also offers a quick and easy environment for testing deployments on a local machine. Users can deploy entire cloud environments in seconds using bundles, which can save a lot of time and effort.

References of Juje:

<https://developer.ibm.com/articles/cl-cloud-orchestration-technologies-trs/>

<https://juju.is/docs/writing-your-first-juju-charm>

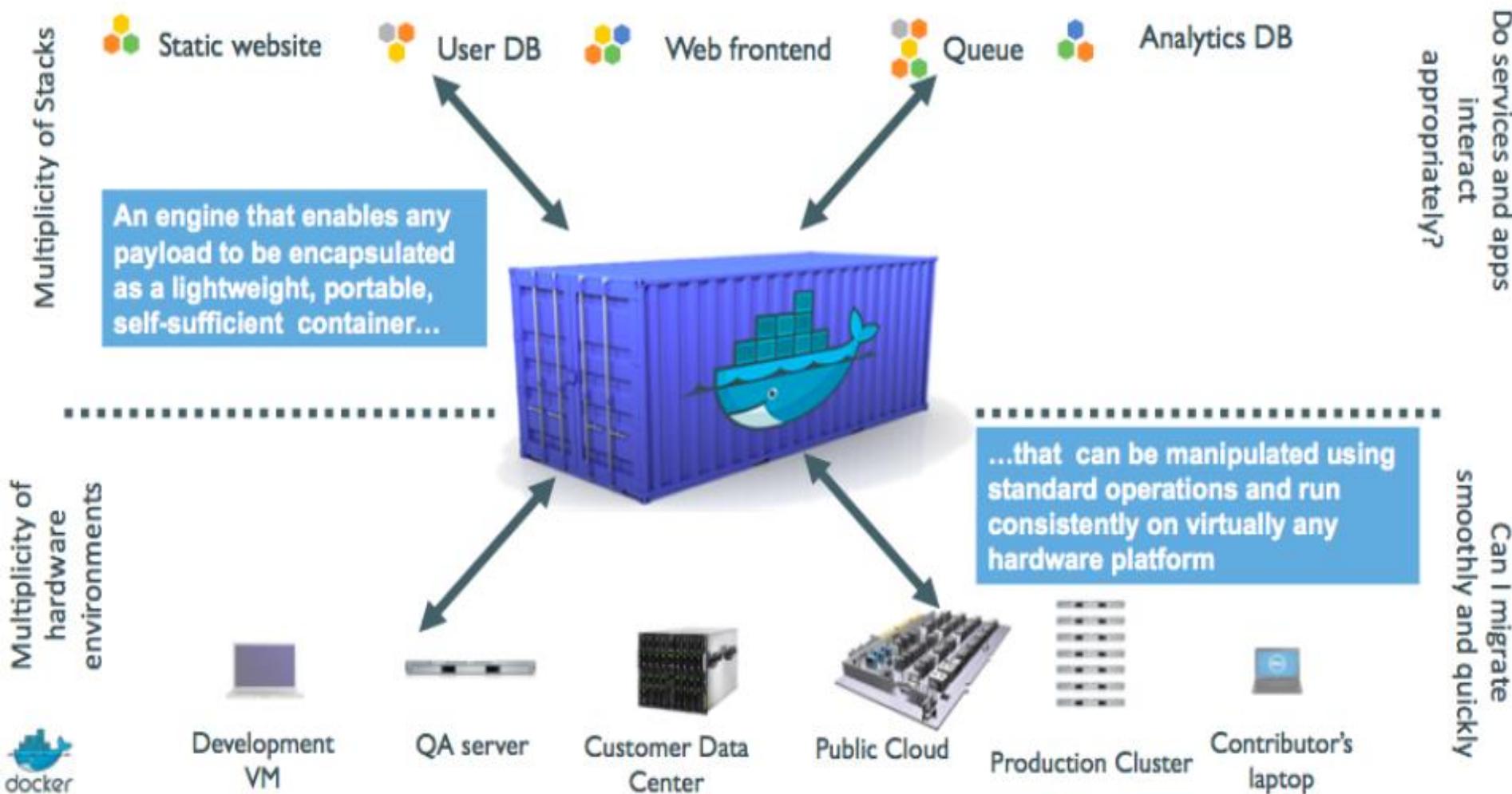
<https://www.youtube.com/watch?v=Lt9-a6pDxsA>

Dockers

- It is an open-source project that automates the deployment of applications inside software containers.
- All applications have their own dependencies, which include both software and hardware resources.
- **Docker is a mechanism that helps in isolating the dependencies per each application by packing them into containers.**
- It provides tools for simplifying DevOps by enabling developers to create templates called images that can be used to create lightweight virtual machines called containers, which include their applications and all of their applications' dependencies.



A shipping container system for applications



What benefits the Docker (Container) provide?

- Docker enables developers to easily **pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere.** Containers do this by enabling developers to isolate code into a single container. This makes it easier to modify and update the program.
- Containerization is an approach of running applications on an OS such that the application is isolated from the rest of the system.
- Main features of Docker are **Develop, ship and run** anywhere.
- They aim at facilitating developers to easily develop applications, ship them along with their dependencies into containers which can then be deployed anywhere. It is aimed to address portability issue.

Note: The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

Ref: <http://www.ce.uniroma2.it/courses/sdcc1819/slides/Docker.pdf> and Also, refer to module 2 slides for more information about Dockers.

Container orchestration (e.g., Docker Swarm)

Container Orchestration

It is the automatic process of managing or scheduling the work of individual containers for applications based on **microservices** within multiple clusters.

Examples of container orchestration platforms are based on open-source versions like **Kubernetes**, **Docker Swarm** or the commercial version from Red Hat **OpenShift**.

Docker container orchestration tool, also known as **Docker Swarm**, can package and run applications as containers, find existing **container** images from others, and deploy a **container** on a laptop, server or cloud (public cloud or private).

Reference: <https://avinetworks.com/glossary/container-orchestration/>

Why Do We Need Container Orchestration?

Container orchestration is used to automate the following tasks at scale:

- Configuring and scheduling of containers
- Provisioning and deployments of containers
- Availability of containers
- The configuration of applications in terms of the containers that they run in
- Scaling of containers to equally balance application workloads across infrastructure
- Allocation of resources between containers
- Load balancing, traffic routing and service discovery of containers
- Health monitoring of containers
- Securing the interactions between containers.

How Does Container Orchestration Work?

- Configurations files tell the container orchestration tool how to network between containers and where to store logs.
- The orchestration tool also schedules deployment of containers into clusters and determines the best host for the container. After a host is decided, the orchestration tool manages the lifecycle of the container based on predetermined specifications. Container orchestration tools work in any environment that runs containers.

Orchestration tools for Docker include the following:

- Docker Machine — Provisions hosts and installs Docker Engine.
- Docker Swarm — Clusters multiple Docker hosts under a single host. It can also integrate with any tool that works with a single Docker host.
- Docker Compose — Deploys multi-container applications by creating the required containers.

Benefits of Containerized Orchestration Tools

Increased portability — Scale applications with a single command and only scale specific functions without affecting the entire application.

- **Simple and fast deployment** — Quickly create new containerized applications to address growing traffic.
- **Enhanced productivity** — Simplified installation process and decreased dependency errors.
- Improved security — Share specific resources without risking internal or external security.

Application isolation improves web application security by separating each application's process into different containers.

Note: Refer to the slides of Module 2 for more information about the Docker files and their creation.

Comparison of cloud Orchestration tools

Chef	Puppet	Heat	Juju	Docker
Mainly used for the automation of deployments. At first, this was used more on OS level for things like deployment of servers, patches, and fixes. Later on, it was used for things like installing middleware.	Originally used at the middleware level for things like installing databases and starting Apache. It explored everything with APIs. In time, its use expanded to installing at the OS level, as well.	Orchestration mechanism from OpenStack.	Pattern-based service layer (automation only) for Ubuntu.	Used as both a virtualization technology and an orchestration tool.
Caters more toward developer-centric operations teams.	Caters to more traditional operations teams with less Ruby programming experience.	Orchestrates everything on OpenStack. Mostly for infrastructure. Heat uses Chef/Puppet for installation.	Works on all popular cloud platforms — Ubuntu local machines, bare metal, etc.	Open platform that enables developers and system administrators to build, ship, and run distributed applications.
Procedural where recipes are written in Ruby code, which is quite natural for developers. Chef's steep learning curve is often seen to be risky in large organizations, where it can be difficult to build and maintain skills in large teams.	Learning curve is less imposing because Puppet is primarily model driven.			
Once you get through the initially steep learning curve, Chef can provide a lot more power and flexibility than other tools.	Puppet is a more mature product with a larger user base than that of Chef.			

Cloud Computing



BITS Pilani

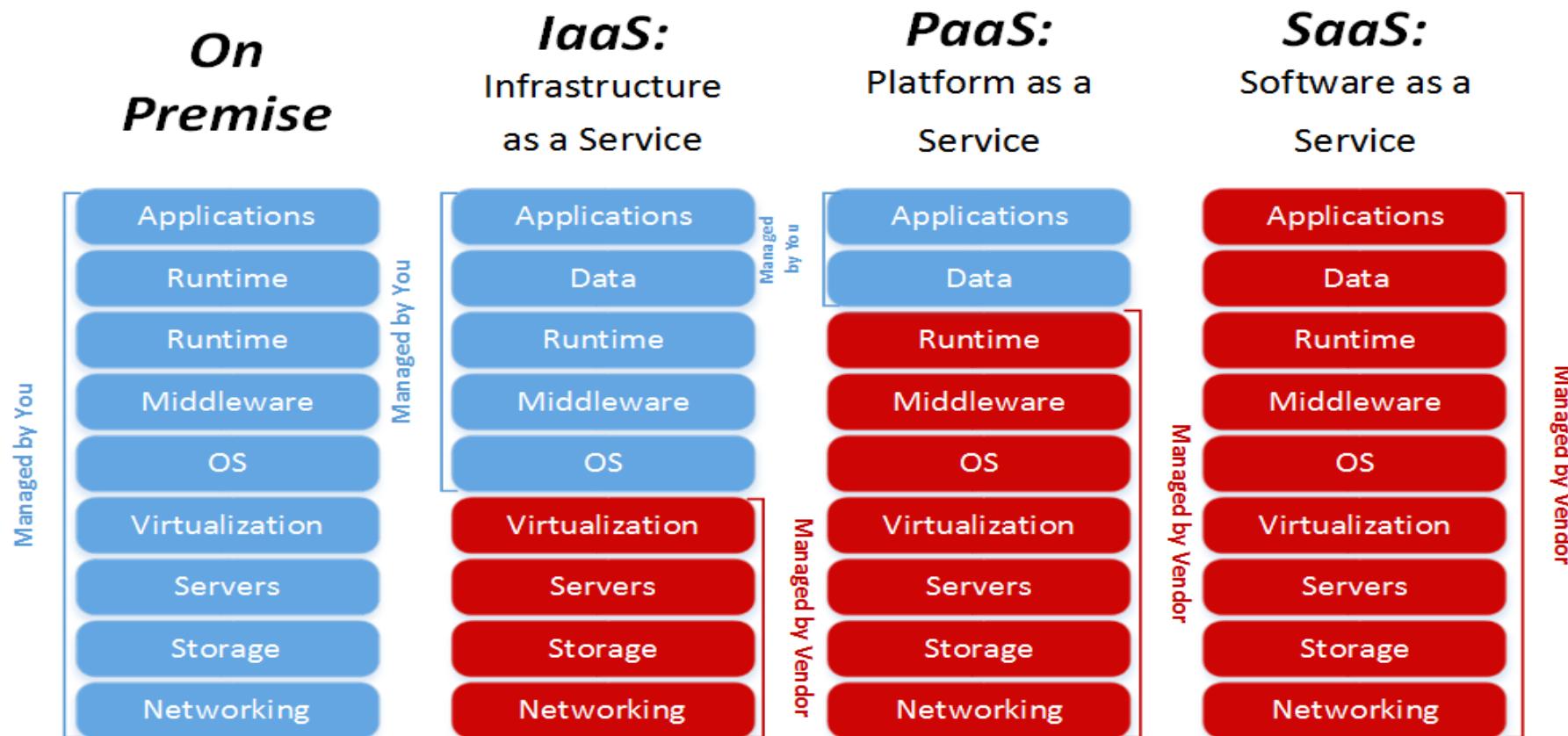
Module 5

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

Agenda

- Platform as a Service (PaaS)
- Building blocks of PaaS
- Characteristics of PAAS
- Windows Azure
- Software as a Service (SaaS)

Dependency on IaaS and PaaS



Introduction to PaaS

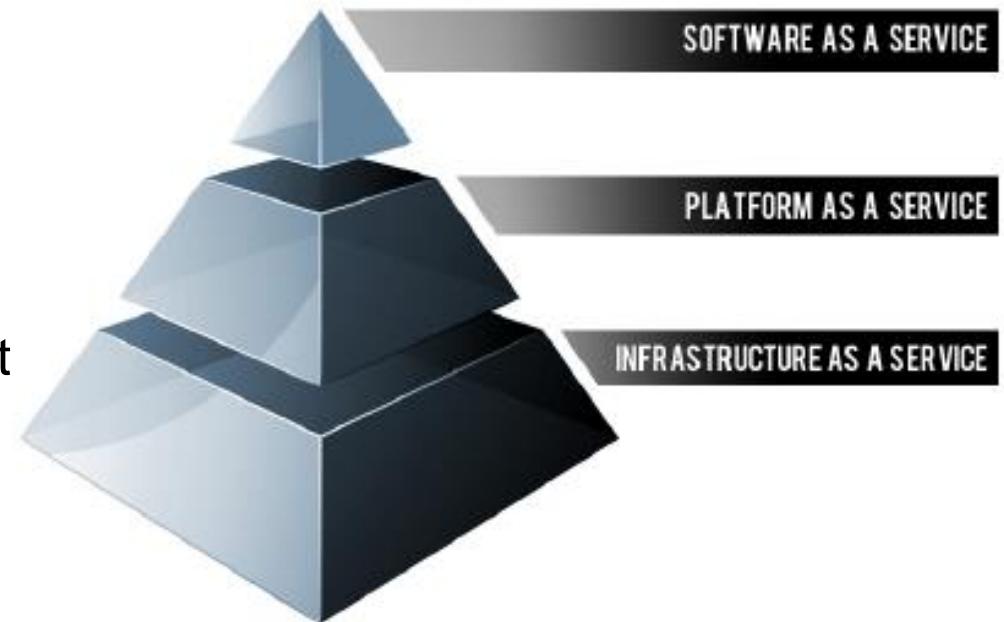
- **PaaS** is a cloud service where the customer gets a **set of application and product development tool hosted on the provider's infrastructure**.
 - The customer can deploy acquired applications or those created using programming languages and tools supported by the provider.
 - PaaS services **are hosted** in the cloud and **accessed by users** simply via their web browser.
 - PaaS services can consist of **preconfigured features that customers can subscribe to**; they can choose to include the features that meet their requirements while discarding those that do not.
 - The customer (developer) here does not control the underlying cloud infrastructure (Servers, OS, security devices,...) but has a control over the deployed applications and also the configurations of hosting environment.
-

Introduction to PaaS (Cont...)

- PaaS systems usually support the complete life cycle of an application
 - Starting from helping in application design
 - PAPIs for application development,
 - Supporting the build and test environment as well as providing the application deployment infrastructure on cloud.
- Additional features during application execution for persistent data usage, state management, session management, versioning and application debugging are also provided by certain PaaS solutions.

Building blocks of PaaS

- Below are some of the features that can be included with a PaaS offering:
 - ❑ Operating system
 - ❑ Server-side scripting environment
 - ❑ Database management system
 - ❑ Server Software
 - ❑ Support
 - ❑ Storage
 - ❑ Network access
 - ❑ Tools for design and development
 - ❑ Hosting



Comparison of In-House Application Development and PaaS

Feature	In-House Application Development	PaaS for Application Development
Multi-Tenancy	Intended for Single or Small group of users	Hundreds to thousands of users, each with multiple active projects. Partition of data is must to protect several users.
User End-Points	Application-based tools, browsers.	Web-browser-based tools
Integrated Development Environment (IDE)	May have several environment and infrastructure for development, test and debugging and production.	Same environment for all phases.
Virtual Machines, Servers, Storage, Databases	Multiple options are available and can be customized to meet any requirement.	Need to work with Infrastructure offered by PaaS providers.
Deployment	Deployment and Scalability are left for installation and go-live phases.	Scalability, failover, and load-balancing are the basic building blocks.
Runtime Monitoring	Development solutions are usually not associated with runtime monitoring.	Built-in monitoring available with the development platform.

Note: PaaS has support of tools to handle billing and subscription management.

Advantages and Risks

Advantages

- Users don't have to invest in physical infrastructure
- PaaS allows developers to frequently change or upgrade operating system features. It also helps development teams collaborate on projects.
- Makes development possible for 'non-experts'
- Teams in various locations can work together
- Security is provided, including data security and backup and recovery.
- Adaptability; Features can be changed if circumstances dictate that they should.
- Flexibility; customers can have control over the tools that are installed within their platforms and can create a platform that suits their specific requirements. They can 'pick and choose' the features they feel are necessary.

Advantages and Risks

Risks

- Since users rely on a provider's infrastructure and software, vendor lock-in can be an issue in PaaS environments.
- Portability/Interoperability.
- Security.
- If a provider stops supporting a certain programming language, users may be forced to change their programming language, or the provider itself. Both are difficult and disruptive steps.

Uses of PaaS

PaaS systems can be used to host a variety of cloud services.

- PaaS can be used by online portal-based applications like Facebook that need to scale to thousands of users.
- Startups who want to host their new application in a Software-as-a-Service model without any upfront cost for hardware or system software, as well as benefit from the flexibility in scaling to a large number of users.
- Enterprises can deploy their Line-of-Business applications in the cloud, taking advantage of the scale and availability while still maintaining security and privacy of data.

Note: PaaS does not typically replace a business' entire infrastructure.

Instead, a business relies on PaaS providers for key services, such as Java development or application hosting.

PaaS Significance through Example Application

- Deploying a typical business tool locally might require an IT team to:
 - To buy and install hardware, operating systems, middleware (such as databases, Web servers and so on) the actual application.
 - To define user access or security.
 - To add the application to existing systems management or application performance monitoring (APM) tools.
- IT teams must then maintain all of these resources over time.

PaaS solution: A PaaS provider supports all the underlying computing and software;— usually through a Web browser interface **users only need to log in and start using the platform e.**

Popular PaaS Providers

- Microsoft Azure
- Google App Engine
- Amazon Web Service (Beanstalk), etc.

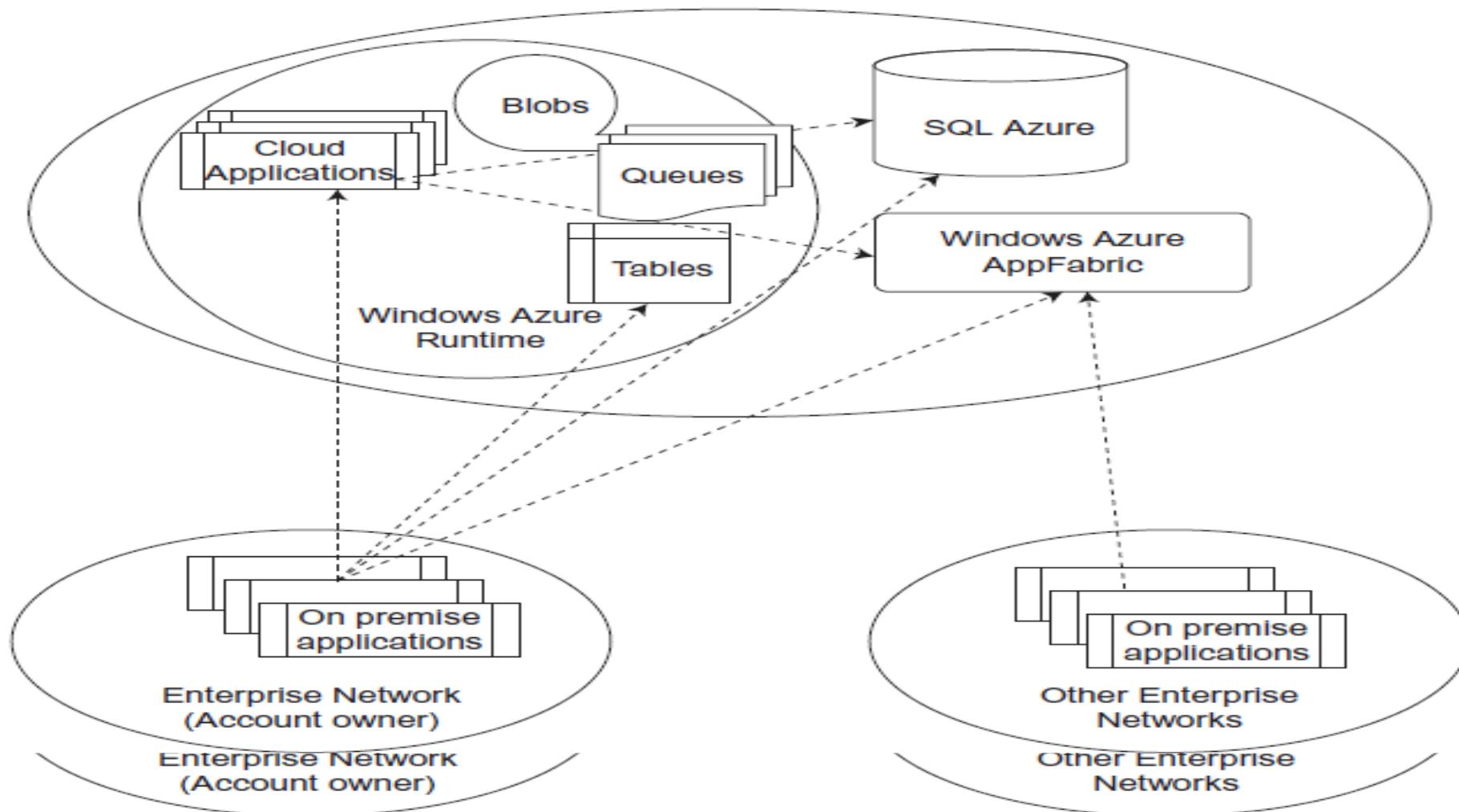
Microsoft Azure Platform

- **Azure** can be thought of as a Cloud Operating System over Microsoft blade servers.
- This operating system handles provisioning, monitoring and complete management of hardware.
- It provides a shared pool of **compute, disk and network resources** to the applications.
- The operating system also manages the application life-cycle on the platform and provides reliable building blocks for authoring applications such as storage, queuing, caching, access control and connectivity services.
- Azure is primarily designed for PaaS capabilities, it also includes certain features for Data-as-a-Service (DaaS) and Infrastructure-as-a-Service (IaaS).

Microsoft Azure Platform (Cont...)

- Individual applications are run in virtual machines that offer a Windows Server compatible environment and are managed by the cloud operating system.
- In addition, the platform offers a relational database (SQL Azure) and a set of services called AppFabric (formerly known as .NET services) that allow on-premise applications to interoperate with applications hosted in the cloud with secure connectivity, messaging, and identity management.

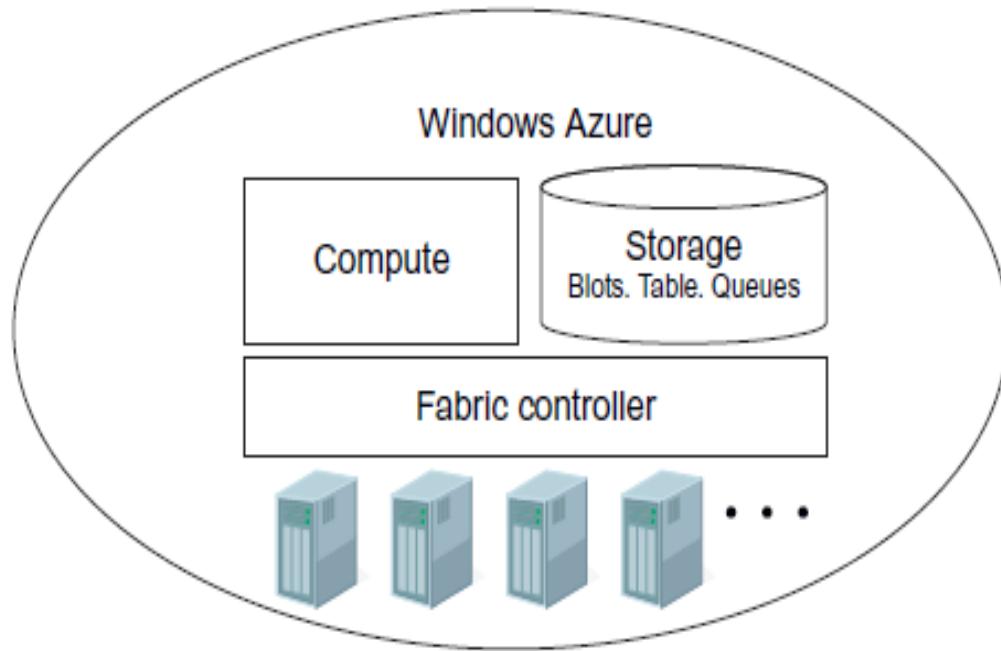
Microsoft Azure Platform



Schematic View of Azure Platfrom Services

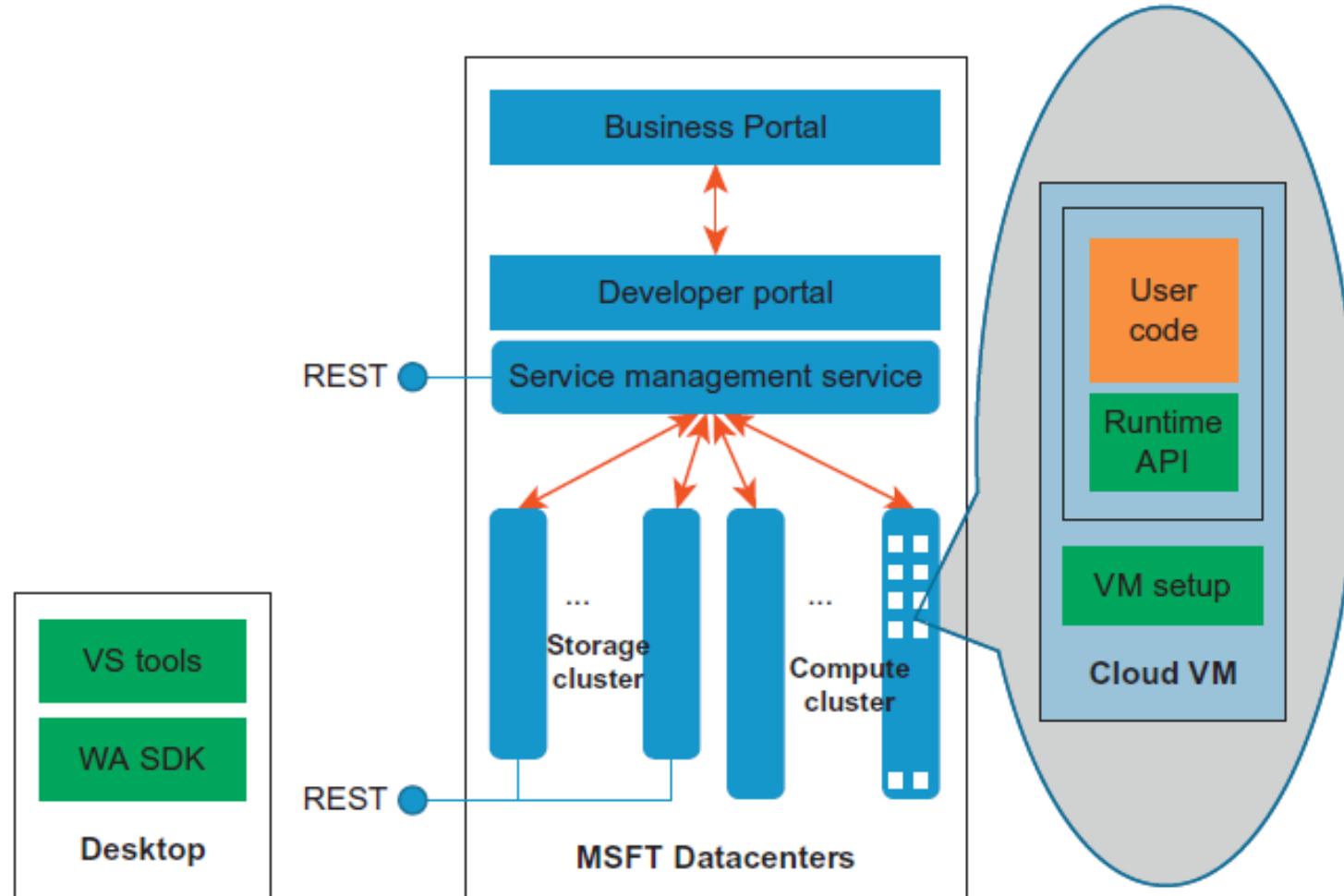
Azure Runtime Environment

- The hosting environment of Azure is called the Fabric Controller.
- It has a pool of individual systems connected on a network and automatically manages resources by load balancing and geo-replication.
- It manages the application lifecycle without requiring the hosted apps to explicitly deal with the scalability and availability requirements.
- Each physical machine hosts an Azure agent that manages the machine – starting from boot up, installation of the operating system and then the application, application monitoring during its execution, and finally even attempting to fix the system if the agent detects any problems



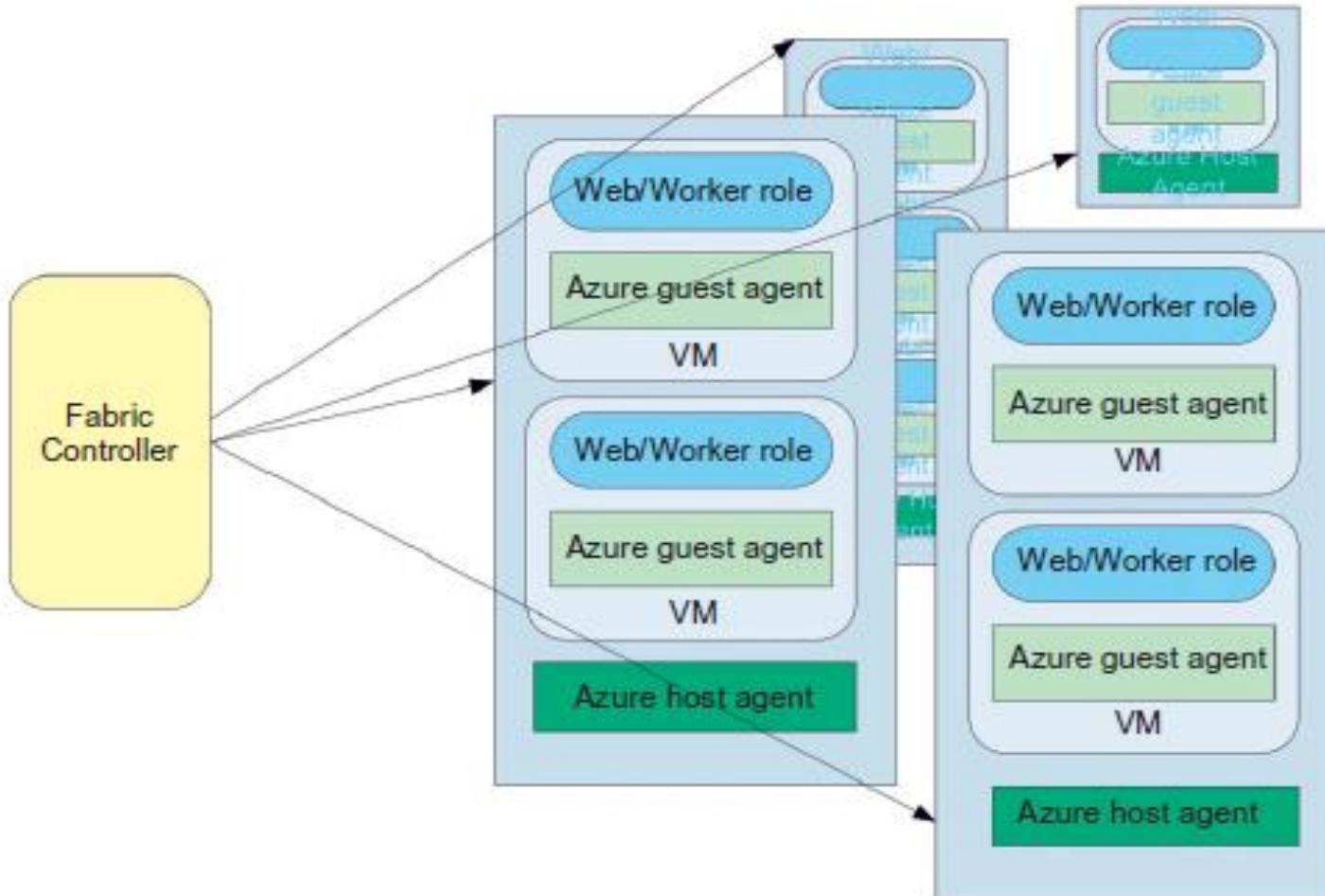
Windows Azure runtime environment components.

Bird's-eye view of the internal modules of the platform.



- At the heart of the system are the storage and compute clusters – vast numbers of machines kept in Microsoft data centers.
- These machines, the operating systems running on them, and applications are managed by the Fabric Controller.
- The external interface of the Azure system consists of a set of **REST APIs** that perform service management and give users access to the storage system.

Fabric Controller Architecture



Fabric Controller

- The **Fabric Controller (FC)** is a distributed program that manages the hardware and applications in a cluster internally used by Azure.
- The key task of the Fabric Controller is **to assign the appropriate resources to an application based on the number of roles, number of role instances, and the upgrade and fault domains specified by the application.**
- Each machine in the cluster runs a hypervisor which hosts virtual machines running Windows compatible OSes.
- The host operating system has an Azure host agent that is responsible for monitoring the health of the physical machine, for starting virtual machine instances, and for reporting the health of the machine to the Fabric Controller.
- The FC monitors host agents through a heart-beat mechanism; if the FC detects that a host hasn't responded to a heartbeat within a pre-determined duration, it considers the machine to be down and takes measures to restore the machine.
- Guest operating systems have a guest Azure agent that monitors the role running on the VM.

-
- The guest agent restarts roles that terminate and keep the host agent informed about the status of the virtual machine.
 - The host agent and the guest agent also communicate through a heartbeat; when the host detects that it hasn't received a heartbeat from a VM, it takes measures to restore the VM.
 - The FC also brings up new machines into the cluster when required, or when existing machines go down for any reason.

SQL Azure

- **SQL Azure** provides a relational database on the cloud. While Azure Table storage service facilitates storing and querying of structured data, it does not provide full-relational capability provided by a traditional relational database management system (RDBMS).
- SQL Azure offers a cloud-based RDBMS system based on SQL Server with nearly all the features offered by on-premise versions of the RDBMS.
- Most applications written for SQL Server will work unchanged when the database is deployed on SQL Azure.
- SQL Azure also frees customers from the operational details of managing large databases. Instead of focusing on service logs, configuration management, and backup, customers can now focus on what matters to their applications: data.

Azure AppFabric

- The **Azure AppFabric is a middleware platform** that developers can use to bridge existing applications/data to the cloud through secure, authenticated connectivity across network boundaries.
- The Azure AppFabric consists of three main components: **the Service Bus, Access Control, and Caching modules.**

Azure AppFabric (Cont....)

- The **Service Bus** provides secure messaging and connectivity between cloud and on-premise applications and data. It exposes on-premise data and services in a secure manner to selected cloud applications **through firewalls, NAT gateways** and other restrictive network boundaries.
- The **Access Control component** provides users across the organizations a single sign-on facility to access services hosted on Azure. Access Control allows an organization to selectively expose its data and services to partners, vendors, and customers in a secure manner, with appropriate authorization at the access points.
- **Caching component** provides an in-memory, scalable, highly available cache for application data, which includes data stored in Azure tables and SQL Azure. Caching improves the performance of both cloud applications and on-premise applications that access cloud resources through intelligent caching of managed objects.

Azure Programming Model

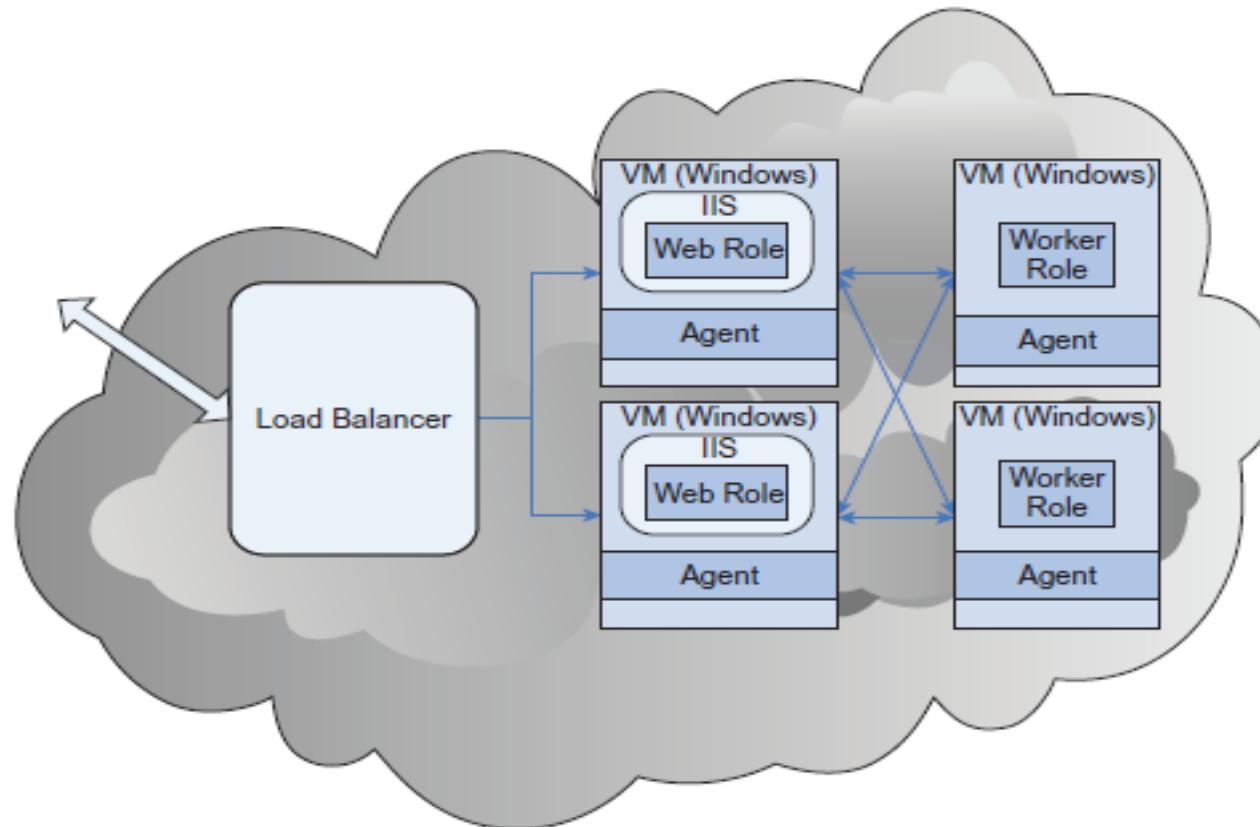
The applications in Azure use the programming model of a **Web role and a Worker role** and applications are required to be configured on Windows Azure with the number of instances of each role that should be created.

- **The Web role** enables web applications to be hosted on IIS servers and **provides external interactivity** while the **Worker role mirrors the heavy-duty processing** that is typically done by Windows services and applications.
- Web roles can be implemented using any technology supported by Microsoft's web-server Internet Information Services (IIS) 7, which includes plain HTML/JS, ASP.NET, and Windows Communication Framework (WCF).

Notes: Windows Azure provides in-built load-balancing to spread requests across Web-role instances of a single application.

- Web roles and Worker roles communicate by either using message queues.

Compute service with two Web roles and two Worker roles



Compute service with two Web roles and two Worker roles.

Azure Cloud Storage Services

1. **Blob service:** For large binary and text data, especially unstructured objects **like images and media.**
2. **Table Service:** For structured storage of non-relational data.
3. **Queue Service:** For message passing between components.
4. **Azure Drives:** To use as mounted file systems.

BLOB

- The Blob service lets applications store and retrieve large binary and text data, called blobs (**Binary Large Objects**). It provides file-level storage and is similar to Amazon S3.
- Blobs are (typically large) unstructured objects **like images and media**.
- It also allows developers to mount blobs as “cloud drives” for programmatic access from within the applications. Windows Azure Drives are used for mounting an NTFS volume to be accessed by an application, and are similar to Amazon EBS.
- Applications deal with blobs as a whole, although they might read/write parts of a blob. Blobs can have optional metadata associated with them in the form of key-value pairs; for instance, an image could have a copyright notice stored as metadata.

BLOB (Cont...)

- Blobs are always stored under containers, which are similar to buckets in Amazon's S3.
- Every storage account must have at least one container, and containers can have blobs within them. Container names can contain the directory separator character ("/") – this gives developers the facility to create hierarchical “file-systems” similar to those on disks.

The blob service defines two kinds of blobs to store text and binary data:**A page blob and a block blob.**

- Page blobs are blobs that are optimized for random read/write operations anywhere in the content of the blob.
- Block blobs are optimized for streaming and are read and written a block at a time.

Note: access to blob and storage services in Azure is through REST interfaces

Table Service

- The Table service is used for structured storage for non-relational data, and is similar to Amazon's SimpleDB.
- For structured forms of storage, Windows Azure provides structured key-value pairs stored in entities known as Tables.
- The Table storage uses a NoSQL model based on keyvalue pairs for querying structured data that is not in a typical database.
- Table is a bag of typed properties that represents an entity in the application domain.

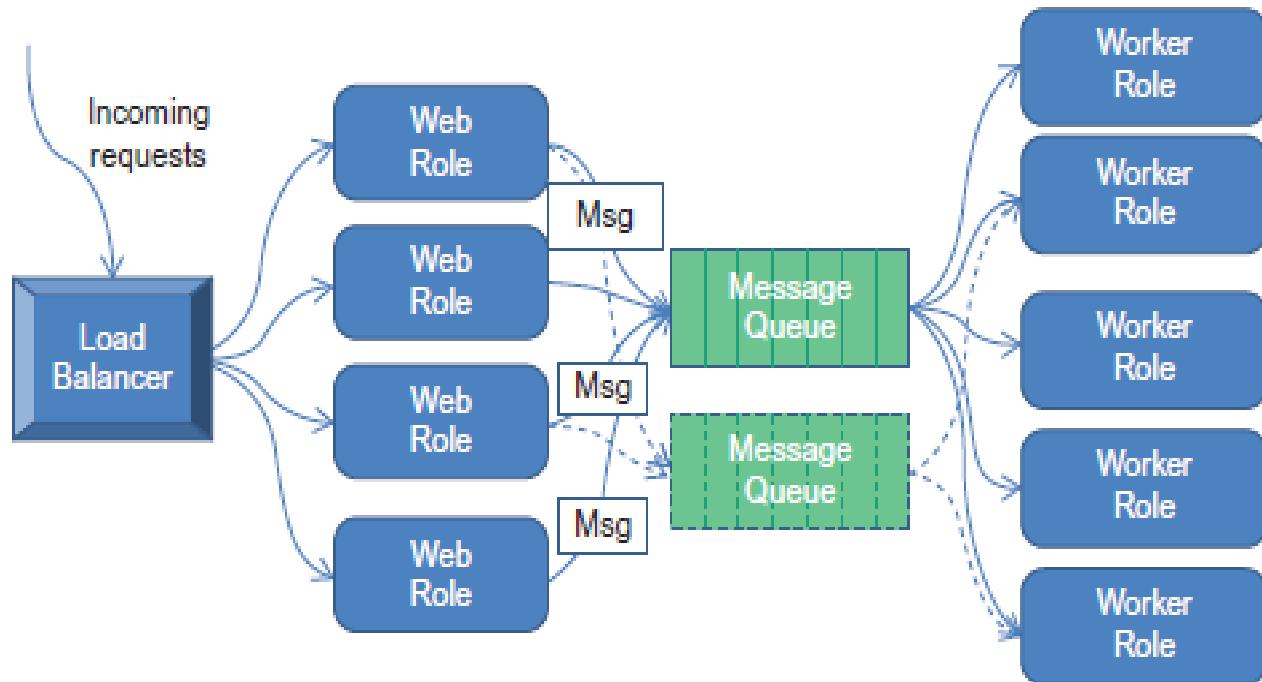
For instance, the definition **{EmployeeId: int, EmployeeName: string}** defines a table that could be used to store (minimal) employee data.

- Data stored in Azure tables is partitioned horizontally and distributed across storage nodes for optimized access.
- Every table has a property called the Partition Key, which defines how data in the table is partitioned across storage nodes – rows that have the same partition key are stored in a partition.
- In addition, tables can also define Row Keys which are unique within a partition and optimize access to a row within a partition. When present, the pair {partition key, row key} uniquely identifies a row in a table.

Queue Service

- Queues are the third kind of storage, **provided for reliable message delivery within and between services.**
- A storage account can have unlimited number of queues, and each queue can store an unlimited number of messages, with the restriction that messages are limited to 8KB (for example).
- Queues are used by Web roles and Worker roles for inter-application communication, and by applications to communicate with each other.

Scalability in Azure



Scaling Web and Worker roles using shared queues.

Note: Each queue offers a throughput of nearly 500 transactions per second. For higher throughput, developers can create multiple queues between Web roles and Worker roles, and have the Worker roles pick up messages either at random, or based on some other scheduling strategy

Example Application “Hello World” using Azure

Objective: To develop a Web application that displays a custom homepage, accepts the visitor’s name, and welcomes him/her.

Important Notes:

- Windows Azure applications typically have multiple instances running on different virtual machines. However, developers need not create nor manage these virtual machines explicitly.
- Developers just write applications either as a Web role and/or Worker role, and tell Windows Azure the number of instances of each role that should be created.
- Windows Azure silently creates the requisite number of virtual machines, and hosts instances of the roles on them.

Development tools:

- Download (**Windows Azure SDK and Visual Studio**) and install development tools required to develop Azure applications.

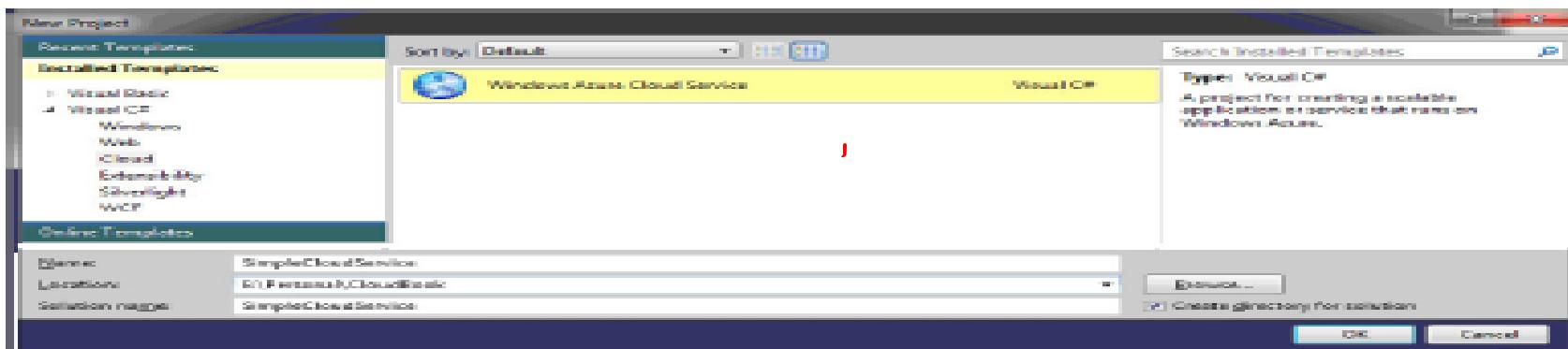


FIGURE 3.1
Creating a new Cloud project in Azure Visual Studio.

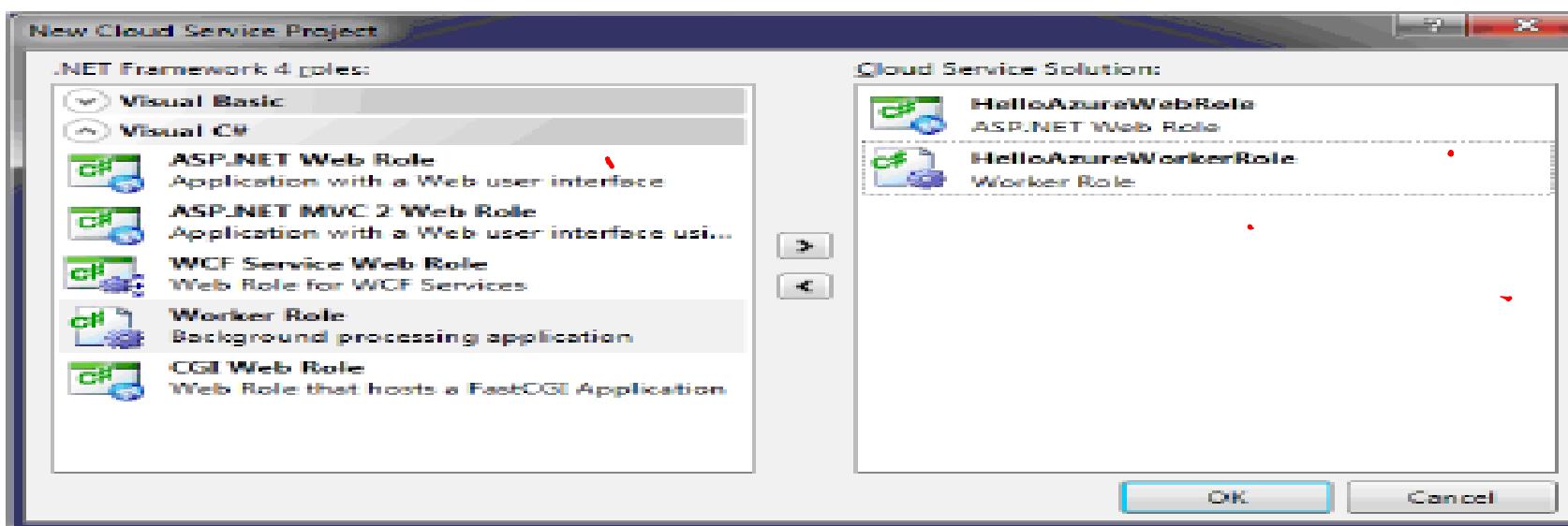


FIGURE 3.2
Choosing Web and Worker roles.

Web Role and Worker Role for “Hello World”

- A Visual Studio Solution is a collection of related projects grouped as logical entities.
- The **Web role is essentially an ASP.NET web application**, with additional features for cloud-based deployment.
- The **Worker role is a simple C# program**. Programming languages such as Python and PhP also could be used to implement worker role.
- Modify the “Default.aspx” page to add a custom message, It is possible to build the solution to create a simple Azure application which consists of two files – **a service package (*.cspkg)** that contains the binaries, and a **service configuration (*.cscfg)** that is used to configure the application.

Message Passing between web role and worker role

How messages can be passed from one component to another?

-For this example, the **Web role will prompt the user** for his/her name **and pass that to the Worker role** and print back the result got from the Web role.

- For this, the handler OnSubmitBtnClick for the “Submit” button needs to pass the input string as a message to the Worker role.
- To enable this communication between the Web role and Worker role, it is necessary to use the message-passing features of Azure.

Message passing through queues

1. To use the message-passing APIs, it is necessary to create a storage account object, and that is specified in the configuration file with the key AzureStorageConnectionString.
2. It is then necessary to get a reference to the queue to transmit messages to the worker. This is needed for the two-way communication between the Worker role and Web role introduced for this example.
 - For this, two queues are used, the first one named **webtoworker** and the second named **workertoweb**. This is accomplished by the GetQueueReference method, which returns a reference to the queue.

Note: The name of the queue

//must be the same at both the web role and the worker role.

3. It is now possible to add a message to the queue for the Worker role – simply send the string typed by the user as the message to the Worker role. This message is added to the **webtoworker** queue. //Wait for a response from the worker
4. The sent message will now reach the Worker role. To process the response from the worker, it is necessary to listen to the messages on the **workertoweb** queue.

Message passing through queues (Cont...)

- Once the message is retrieved, delete the message from the queue, so that the same message is not processed twice, and show the reply to the user by setting the ResultLabel field to the text returned by the Worker role.

Note1: The Worker role has to monitor the incoming message queue for new messages, and when it gets a message, it has to respond back with a message, say, the server time.

Note2: a ConfigurationSettingPublisher must be set for both the Web role and Worker role so that the correct configuration files are read.

Azure Test and Deployment

- Visual Studio must be started **in administrator mode** to debug cloud applications.
- The application would be now running on the local machine on top of a Windows Azure emulator known as the development fabric.
- The development fabric provides the functionality of Windows Azure on the local machine so that users can debug the logic of their services in the familiar environment provided by Visual Studio.

Deploying Azure applications:

- one needs to obtain an Azure subscription from the Microsoft Online Customer Portal.
- Once a subscription is created, a developer login is available to the Windows Azure developer portal at <http://windows.azure.com>, where the developer can create a new project.
- Once a subscription has been obtained in the developer portal, **the next step is to create a hosted service and a storage account for the new application.**
 - **The hosted service** creates the end points for accessing the application. The unique service URL that is selected here will be used by clients to access the application.
 - Similarly, the storage account exposes the endpoints for accessing cloud storage for the application.

Deployment environments

-Two deployment environments – production and staging.

- Both environments host the application on the cloud, but the **staging environment** creates the deployment with a **temporary URL** as the end point, while **the production deployment** uses the **service URL**.
- Applications can be moved from production to staging and vice-versa at any time.
- The staging application is hosted on a temporary URL, one that is nearly impossible to discover accidentally. **This allows developers to deploy their applications on the cloud and test them before making them publicly available.**
- Finally, the deployment switch icon is selected to move the service into production.

The application would be now available on, e.g., <http://simpleazureservicecc.cloudapp.net>.

Note: If more instances of the Web or the Worker role are needed, the service configuration from the configuration page in the portal can be modified.

Summary of Deploying Applications on Azure

In general, the following are the steps needed to build an Azure application.

1. First the Web and Worker roles for the new application were created and tested in the development environment.
2. Then an Azure subscription needs to be obtained and use it to create a hosted service and storage account.
3. Next, the application would be tested in the development environment but using cloud storage.
4. Further testing on the cloud, but in a staging environment.
5. In the end, the staging environment was switched to production making the application globally accessible at the desired URL.

Billing on Azure

CPU Time Billing

Windows Azure starts billing for CPU time when applications are deployed in the staging environment and continues billing when applications (either in staging or production) are suspended. Applications must be deleted from the portal for billing to stop.

Google App Engine

- Google App Engine (GAE) is an example of a Platform as a Service (PaaS) that enables users to develop and host their own applications.
- It enables us to run web applications on Google Infrastructure.
- Application on GAE are easy to build, maintain and scale as the traffic and storage needs grow.
- It also provides distributed storage with replication and load balancing of client requests.

Features of Google AppEngine

Feature	Description
Popular Languages	Node.js, Java, Ruby, C#, Go, Python and PHP
Fully managed	Focus on Code while App Engine manages infrastructure concerns
Powerful application diagnostics	To monitor the health and performance of our app and diagnose and fix bugs errors easily
Application versioning	Different versions of app can be hosted and easily create development, test, staging, and production environments
Application security	Help safeguard our application by defining access rules with AppEngine firewall and leverage managed SSL/TLS certificates by default at no addition cost
Services ecosystem	Tap a growing ecosystem of Google Cloud services from our app including excellent suit of cloud developer tools

Types of Scaling

- Basic
 - Creates instances when your application receives requests.
 - Each instance will be shut down when the application becomes idle.
- Automatic Scaling
 - Creates instance based on request rate, response latencies, or other application metrics that you specify.
- Manual Scaling
 - Allows you to manually specify the number of instances that continuously run regardless of the load level.

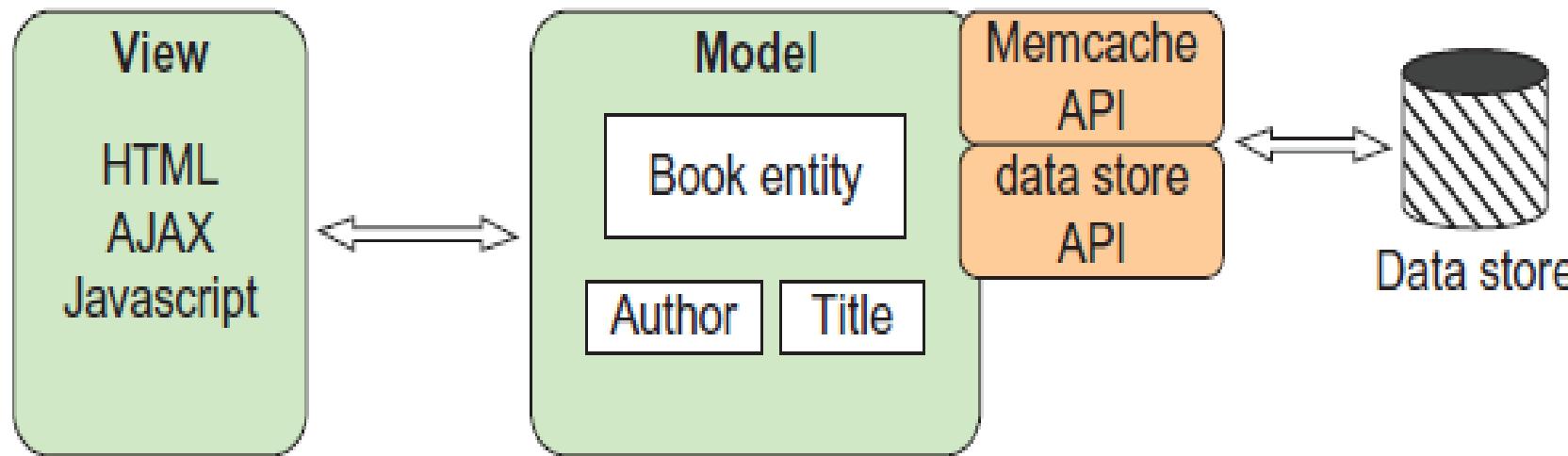
App Engine Security

- Applications run in a secure environment that provides limited access to the underlying operating system
- **Sandbox** isolates the application that is independent of hardware, operating system and physical location of a web server.
- Sandbox restricts what our application can do
- **Limitations imposed by sandbox (for security):**
 - An application can only access other computers over internet using the provided URL fetch and email services.
 - Applications cannot write to local file system.
 - Application code runs only in response to a web request, a queued task or a scheduled task and must return the response data within 60 seconds.
 - A request handler cannot spawn a subprocess or execute code after the response has been sent.

Benefits of App Engine for WebSites

- Availability
- Faster Launch Time
- Easy to use
- Diverse set of APIs (Searching, Application log files, Cloud storage, SSL support, URL Fetch, Mail, etc.)
- Scalability
- Smart Pricing model

Persistent Storage



Using persistent stores in Google App Engine.

Datastore

- The datastore service provides a distributed data storage with a query engine that supports transaction semantics.
- The **datastore is a key-value storage** (similar to Amazon SimpleDB)
- Every data record is an entity and is identified using a key and a set of properties. Operations on groups of entities can also be performed if a transaction requires it.
- The App Engine datastore provides high availability by replicating the data in multiple copies and providing a well-proven algorithm (called Paxos algorithm) to synchronize the multiple copies and provide eventually consistent responses

memcache

- The memcache service can be used to speed up the datastore queries by having a local cache.
For example, if a book is newly published and has become a hot seller, instead of going to the datastore for updating the sales data, the developer may wish to keep that entity information in cache, update it locally and write back to datastore later.
- Similarly, when multiple clients are requesting the same, it helps to serve the response from the cache instead of a datastore.

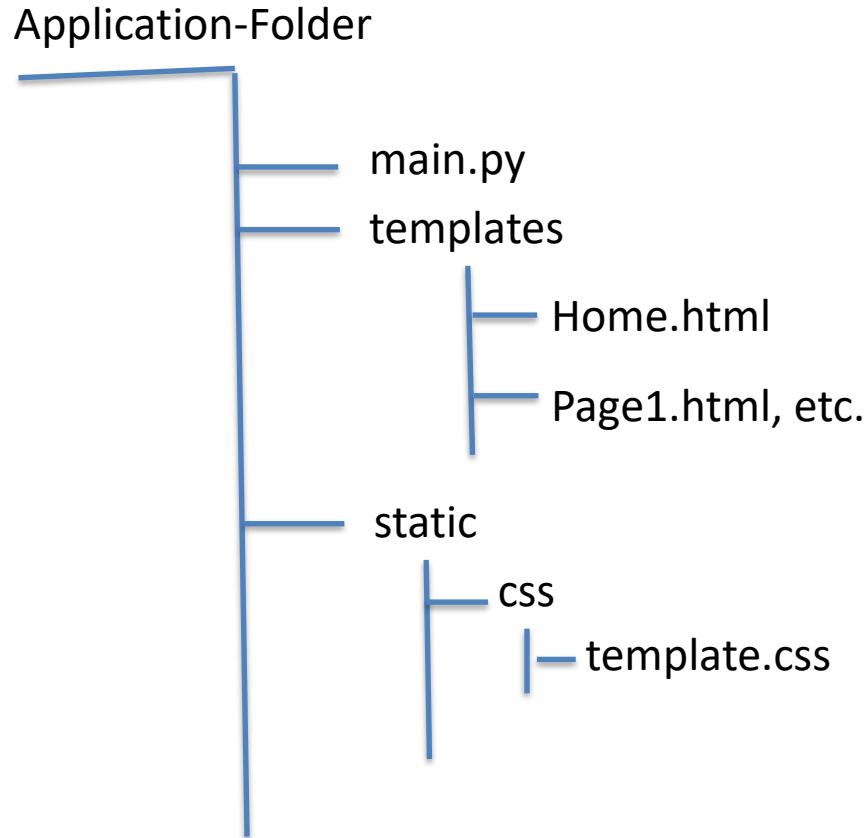
App Engine SDK (Software Development Kit)

- The App Engine SDK allows us to run Google App Engine applications on our local computer. It simulates the run-time environment of the google app engine infrastructure.
- A separate SDK needs to be downloaded and installed as per the programming language.
- The SDK can be used for developing, deploying and managing applications.
- The SDK comes with a local web server for test deployment. We can then deploy the developed application onto the Google App Engine.
- This local web server simulates the secure runtime or App Engine sandbox environment with limited access to the underlying operating system.

For example, the application can only be accessed using HTTP on specific ports. It cannot write to the file system and can read only files that were uploaded along with application code.

Developing a Web Application using Flask Framework

Directory Structure



Installations required:

- Python 3
- Flask
- SDK

Choosing an App Engine environment

- We can run your applications in App Engine using the **flexible environment** (Application instances run **within a Docker container**) or **standard environment** (Application instances run within a **sandbox**) .
- We can also choose to simultaneously use both environments for your application and allow your services to take advantage of each environment's individual benefits.

The standard environment is optimal for applications with the following characteristics:

- Source code is written in **specific versions of the supported programming languages**:
 - Python 2.7, Python 3.7, Python 3.8
 - Java 8, Java 11
 - Node.js 8, Node.js 10, Node.js 12, and Node.js 14 (preview)
 - PHP 5.5, PHP 7.2, PHP 7.3, and PHP 7.4 (preview)
 - Ruby 2.5, Ruby 2.6 (preview), and Ruby 2.7 (preview)
 - Go 1.11, Go 1.12, Go 1.13, and Go 1.14
- Intended to **run for free or at very low cost**, where you pay only for what you need and when you need it. For example, your application can scale to 0 instances when there is no traffic.
- Experiences **sudden and extreme spikes of traffic** which require immediate scaling.

The flexible environment is optimal for applications with the following characteristics:

- Source code that is written in a version of any of the supported programming languages:
Python, Java, Node.js, Go, Ruby, PHP, or .NET
- Runs in a Docker container that includes a custom runtime or source code written in other programming languages.
- Uses or depends on frameworks that include native code.
- Accesses the resources or services of your Google Cloud project that reside in the **Compute Engine network**.

Demonstration of a Simple Hello World Application (Python based) using Google AppEngine

- **Create a project:** Projects bundle code, VMs, and other resources together for easier development and monitoring.
- **Build and run your "Hello World!" app:** Develop and deploy the app to the web using the **gcloud** command.

URL: <https://cloud.google.com/community/tutorials/python-gae-quickstart>

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

1. Log into the Google Cloud Platform (GCP) account.
2. Select **AppEngine** on Google Cloud Platform
3. Click **CreateApplication**, select the **Region** (e.g., US Central)
4. Then, click **Create app** and wait for a while, **application environment would be created.**

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

Project Setup:

Google Cloud organizes resources into projects, which collect all of the related resources for a single application in one place. For Project Details: (https://cloud.google.com/resource-manager/docs/creating-managing-projects#creating_a_project)

Using Cloud Shell: Cloud Shell is a built-in command-line tool for the console. We will use Cloud Shell to deploy our hello world app.

1. Open Cloud Shell: Open Cloud Shell by clicking the **view gcloud command** button in the navigation bar in the upper-right corner of the console on Google Cloud Instance and then select **Run in Cloud Shell**.

2. Clone the sample code:

Use Cloud Shell to clone and navigate to the "Hello World" code. The sample code is cloned from your project repository to the Cloud Shell.

Note: If the directory already exists, remove the previous files before cloning.

In Cloud Shell, enter the following:

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples
```

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

3. Then, switch to the **hello world** directory:

```
cd python-docs-samples/appengine/standard_python3/hello_world
```

4. Configuring our deployment

We are now in the main directory for the sample code. It contains the following files:

- **app.yaml** (this file contains the minimal amount of configuration required for a Python 3 application. The **runtime** field specifies the python37 run-time environment.)
- **main.py**
- **main_test.py**
- **requirements-test.txt**
- **requirements.txt (list of software's to be installed)**

We can check the content of each of the above files on cloud shell, e.g., **cat app.yaml**

Notes:

- The application is a simple Python application that uses the Flask web framework. This Python app responds to a request with an HTTP header and the message Hello World!.
- App Engine uses YAML files to specify a deployment's configuration. app.yaml files contain information about our application, like the **runtime environment, environment variables**, and more.

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

5. Testing our app

Cloud Shell lets us test our app before deploying to make sure it's running as intended, just like debugging on our local machine.

- To test your app, first create an isolated virtual environment. This ensures that your app does not interfere with other Python applications that may be available on your system.

```
virtualenv --python python3 ~/envs/hello_world
```

- Activate your newly created virtual environment:

```
source ~/envs/hello_world/bin/activate
```

- Use pip to install project dependencies. This "Hello World" app depends on the Flask microframework:

```
pip install -r requirements.txt
```

- Finally, run your app in Cloud Shell using the Flask development server:

```
python main.py
```

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

```
* Serving Flask app "main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 308-797-272
```

Note1: We can Preview our app with "**Web preview**" button available on the right corner of the cloud shell window and then select **Preview on Port 8080**.

Note2: Terminate the instance of the application by pressing **Ctrl+C** in the Cloud Shell.

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

6. Deploying to App Engine

```
gcloud app deploy
```

If you face some issue due to the already existing applications, then follow the below:

1. **gcloud projects list** (it will show all the projects available, i.e., Project_Id, Name and Project_Number)
2. Then we may set it for your current workspace by running:

gcloud config set project VALUE (Project_Id has be specified in place of VALUE.) and
then type **gcloud app deploy**

It will return the URL after successful deployment. This URL can be used to access the deployed application.

Note: The default URL of your app is a subdomain on appspot.com that starts with your project's ID: .appspot.com.

Demonstration of a Simple Hello World Application using Google AppEngine (Cont...)

7. Disable your project

- Go to the **Settings** page of App Engine.
- Click **Disable Application**.

Creating New Project

- Go to cloud console
- Create a new project using the below command:

gcloud projects create ***PROJECT_ID***

Where ***PROJECT_ID*** is the ID for the project you want to create. A project ID must start with a lowercase letter, and can contain only ASCII letters, digits, and hyphens, and must be between 6 and 30 characters.

Pricing

- AppEngine has a pricing model that scales with your app's usage.
- Pricing is different for apps in the standard and flexible environments.

Standard environment:

- Apps in the standard environment have a [free tier](#) for App Engine resources.

- Accrual of instance hours begins when an instance starts and ends as described below, depending on the type of scaling we specify for the instance:
 - **Basic or automatic scaling:** accrual ends fifteen minutes after an instance finishes processing its last request.
 - **Manual scaling:** accrual ends fifteen minutes after an instance shuts down.

Instance class	Cost per hour per instance
B1	\$0.058
B2	\$0.116
B4	\$0.232
B4_1G	\$0.348
B8	\$0.464
F1	\$0.058
F2	\$0.116

Pricing (Cont...)

Network resources:

Resource	Unit	Unit cost (in US \$)
Outgoing network traffic*	Gigabytes	\$0.12
Incoming network traffic	Gigabytes	Free

Legacy App Engine resources:

Resource	Unit	Unit cost (in US \$)
Blobstore stored data*	Gigabytes per month	\$0.026
Dedicated memcache	Gigabytes per hour	\$0.083
Logs API	Gigabytes	\$0.138
Search API**: Total storage (documents and indexes)	per GB per month	\$0.207
Search API: Queries	per 10K queries	\$0.575
Search API: Indexing searchable documents	per GB	\$2.30
Sending email, shared memcache, cron, APIs (Task Queues, Image, Files, Users)		No Additional Charge

Flexible environment price details: <https://cloud.google.com/appengine/pricing>

Reference for Python Web Application

<https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/>

SaaS (Software As A Service)

The agenda of SaaS:

- ❑ What is SaaS?
- ❑ Traditional Model
- ❑ How is it delivered?
- ❑ SaaS Architecture
- ❑ Advantages of SaaS
- ❑ User and Vendor benefits of SaaS

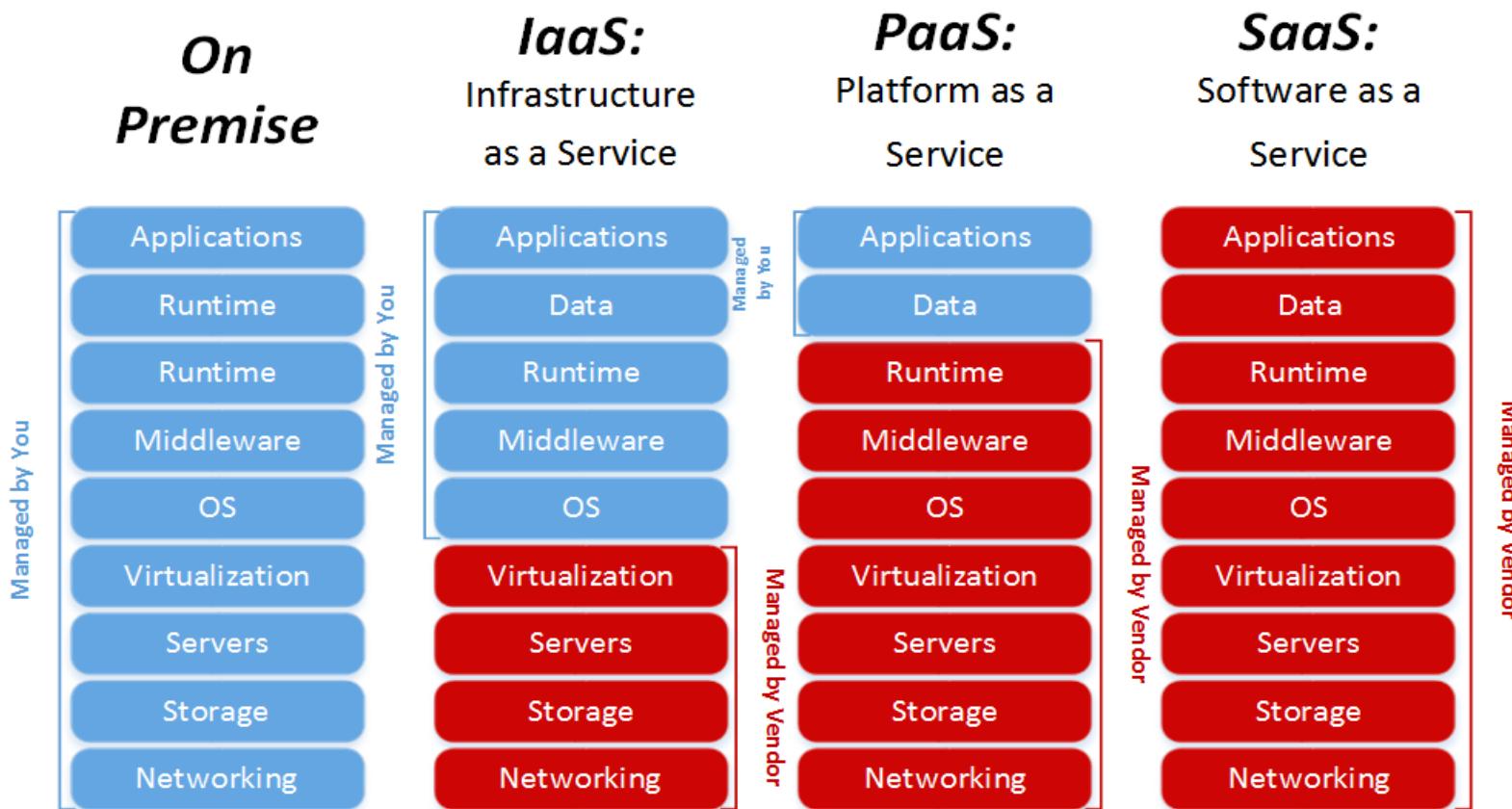
Software As A Service (SaaS)

IaaS: It delivers computing resources as a service.

PaaS: Application deployment platforms can be delivered as a service. These are generic services that can be used by developers for creating new applications

SaaS: It delivers application software as a service.

Dependency on IaaS and PaaS

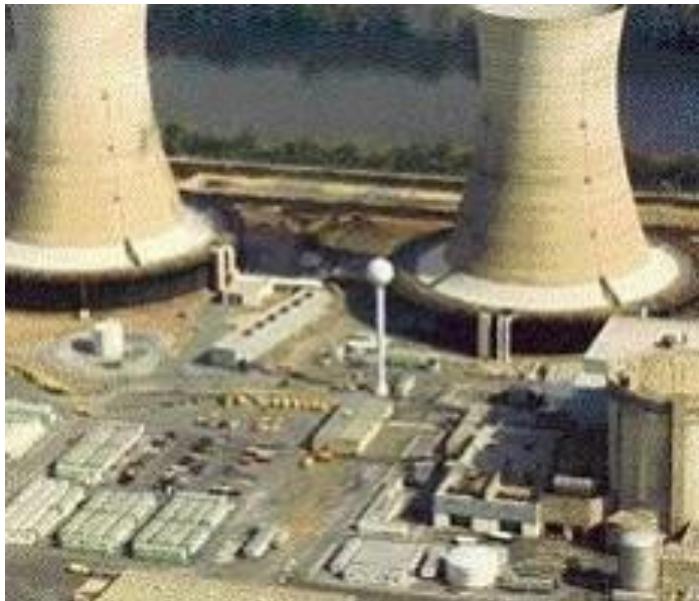


Problems in traditional Model

- In the traditional model of software delivery, the customer acquires a perpetual license and assumes responsibility for managing the software.
- There is a high upfront cost associated with the purchase of the license, as well as the burden of implementation and ongoing maintenance.
- ROI is often delayed considerably, and, due to the rapid pace of technological change, expensive software solutions can quickly become obsolete.

Problems in traditional Model

Traditional Software



Build Your Own

On-Demand Utility

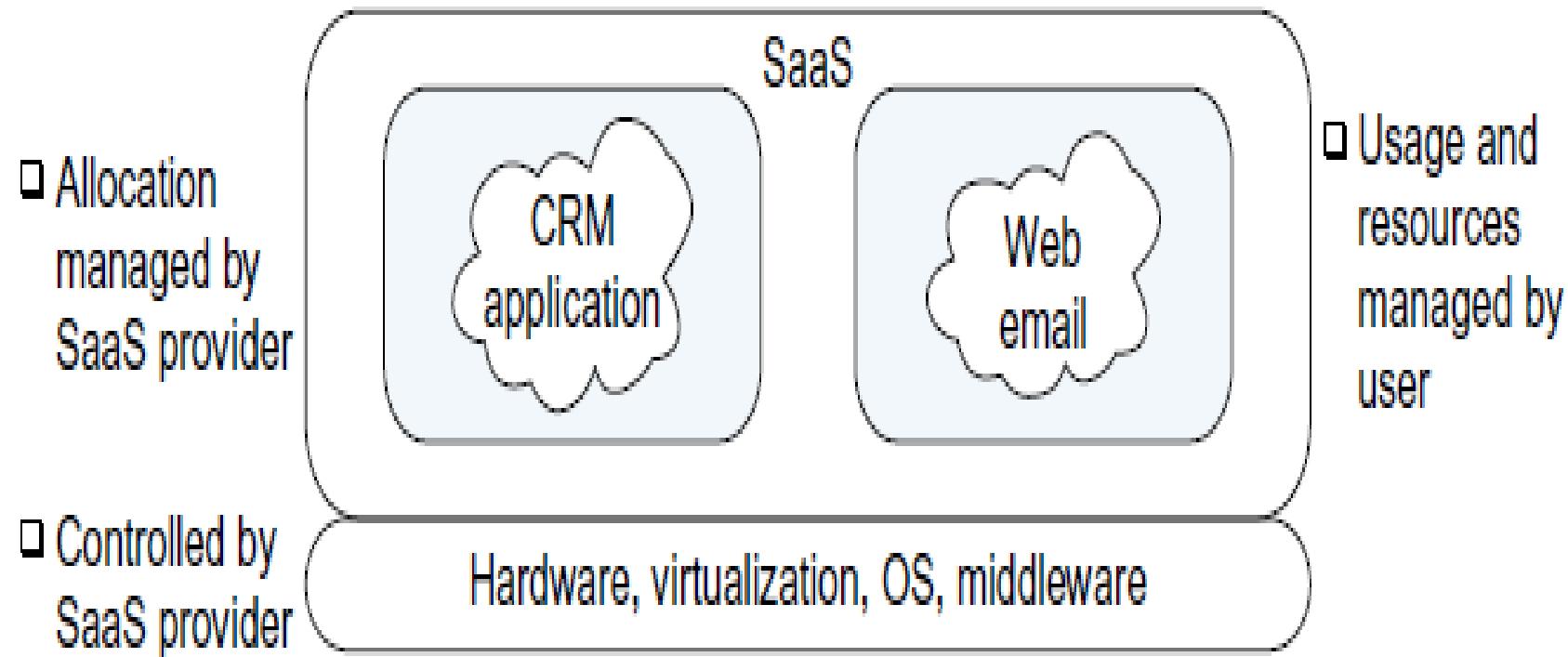


**Plug In, Subscribe
Pay-per-Use**

What is SaaS?

- It is a complete cloud computing service model in which the computing hardware and software, as well as the solution itself, are provided by a vendor as a complete service offering. It is referred to as the Software as a Service (SaaS) model.
- SaaS can be described as software that **is deployed on a hosted service** and can be **accessed globally over the Internet, most often in a browser**.
- Every computer user is familiar with SaaS systems, which are either replacements or substitutes for locally installed software.
- Examples of SaaS software for end-users are Google Docs, Gmail and Calendar, YouTube, etc.

SaaS cloud Model



SaaS Characteristics

1. The software is available over the Internet globally through a browser on demand.
2. The typical license is **subscription-based or usage-based** and is billed on a recurring basis. In a small number of cases a flat fee may be charged, often coupled with a maintenance fee.
3. The software and the service are monitored and maintained by the vendor, regardless of where all the different software components are running.
There may be executable client-side code, but the user isn't responsible for maintaining that code or its interaction with the service.
4. Reduced distribution and maintenance costs and minimal end-user system costs generally make SaaS applications cheaper to use than their shrink-wrapped versions.

SaaS Characteristics (Cont...)

5. Automated upgrades, updates, and patch management and much faster rollout of changes.
6. SaaS applications often have a much lower barrier to entry than their locally installed competitors, a known recurring cost, and they scale on demand (a property of cloud computing in general).
7. All users have the same version of the software so each user's software is compatible with another's.
8. SaaS supports multiple users and provides a shared data model through a single-instance, multi-tenancy model.

SaaS – How is it delivered

- The SaaS based software is delivered to the users with the help of internet.
- Traditional desktop applications such as word processing and spreadsheet can now be accessed **as a service in the Web**.
- This model of delivering applications, alleviates the burden of software maintenance for customers and simplifies development and testing for providers.

Customization Support for SaaS

- Some SaaS solutions expose Application Programming Interfaces (**API**) to **developers to allow them to create custom composite applications.**
- These APIs may alter the security model used, the data schema, workflow characteristics, and other fundamental features of the service's expression as experienced by the user.
- Examples of an SaaS platform with an exposed API are **Salesforce.com** and **Quicken.com**.

SaaS Application Architecture

Scaling the application:

- Maximizing concurrency, and using application resources more efficiently, i.e. optimizing locking duration, sharing pooled resources such as threads and network connections, caching reference data, and partitioning large databases.

SaaS Application Architecture (Cont...)

Multi-tenancy :

- One application instance must be able to accommodate users from multiple other companies at the same time
- All transparent to any of the users.
- This requires an architecture that maximizes the sharing of resources across tenants and yet should be able to differentiate data belonging to different customers.

SaaS Application Architecture

Configurable:

- To customize the application for one customer will change the application for other customers as well.
- Traditionally customizing an application would mean code changes
- Each customer uses metadata to configure the way the application appears and behaves for its users.
- Customers configuring applications must be simple and easy without incurring extra development or operation costs

Advantages of SaaS for Users

- Users can directly use these applications without any new software installations on their local machines, as these applications will now execute within a web browser (YouTube, for example).
- The required software will be available on multiple platforms and can be used from any of the user's devices (say, home PC, office PC, mobile device).
- If the application is needed only for a short period of time, the user can simply pay per use.
- Applications can be customized for different users both in terms of user interface and selected features, so there is no loss in flexibility.

Advantages of SaaS from Vendors' Point of View

- Many service providers can economically offer a new application as a service directly instead of creating packages and distribution channels.
- It helps to ensure that the software is not pirated.
- The vendors need not worry about distributing updates of newer versions of the application; only the cloud application needs to be changed and the new version will now be used the next time the consumer accesses it.

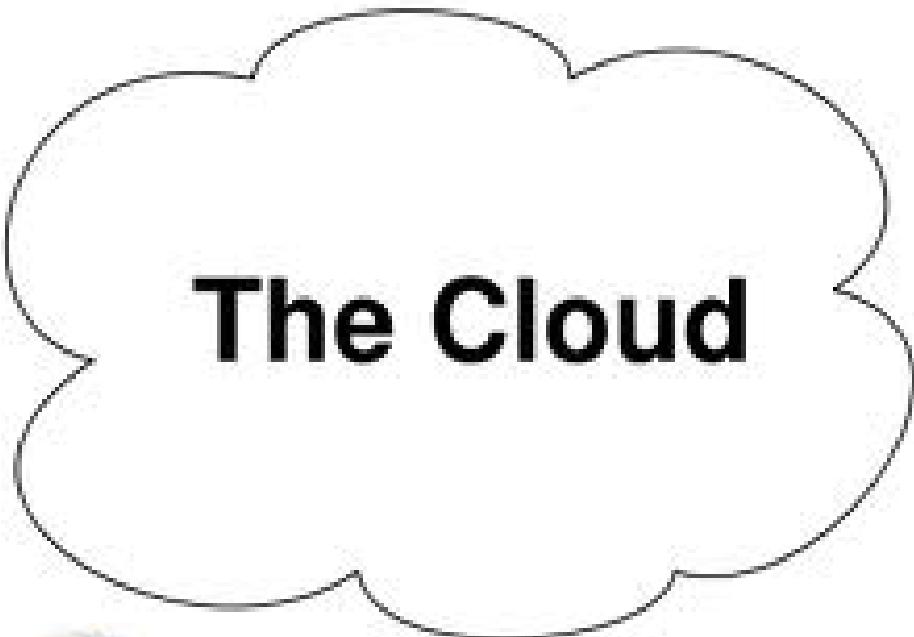


Module 6



BITS Pilani

Cloud ? That looks easy!!!



Is it So?????



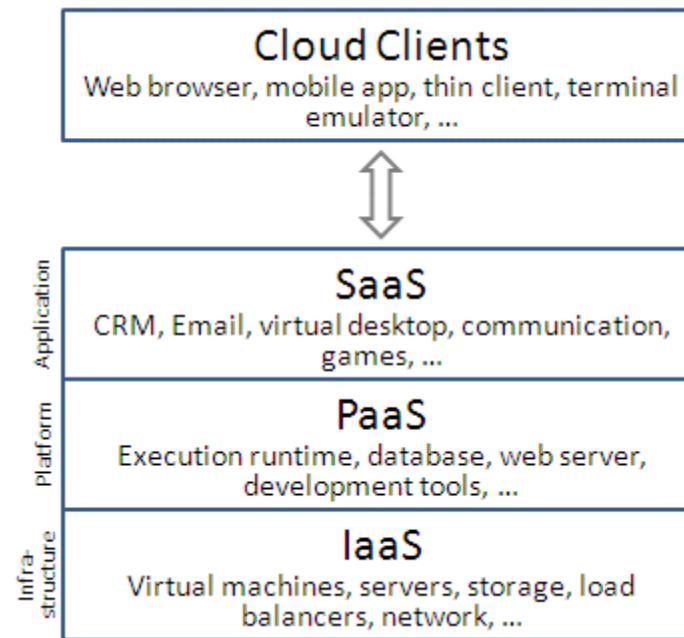
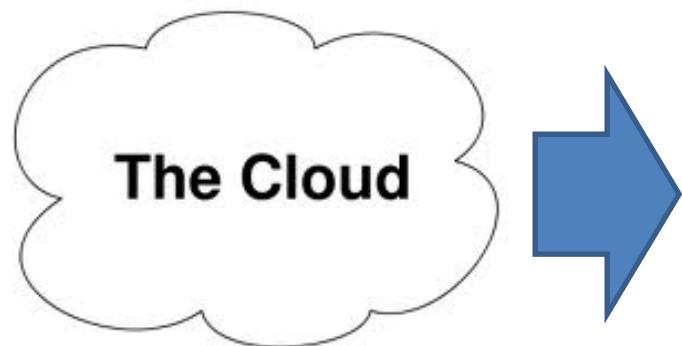
Introduction



So CLOUD requires managing.....



But what will you manage

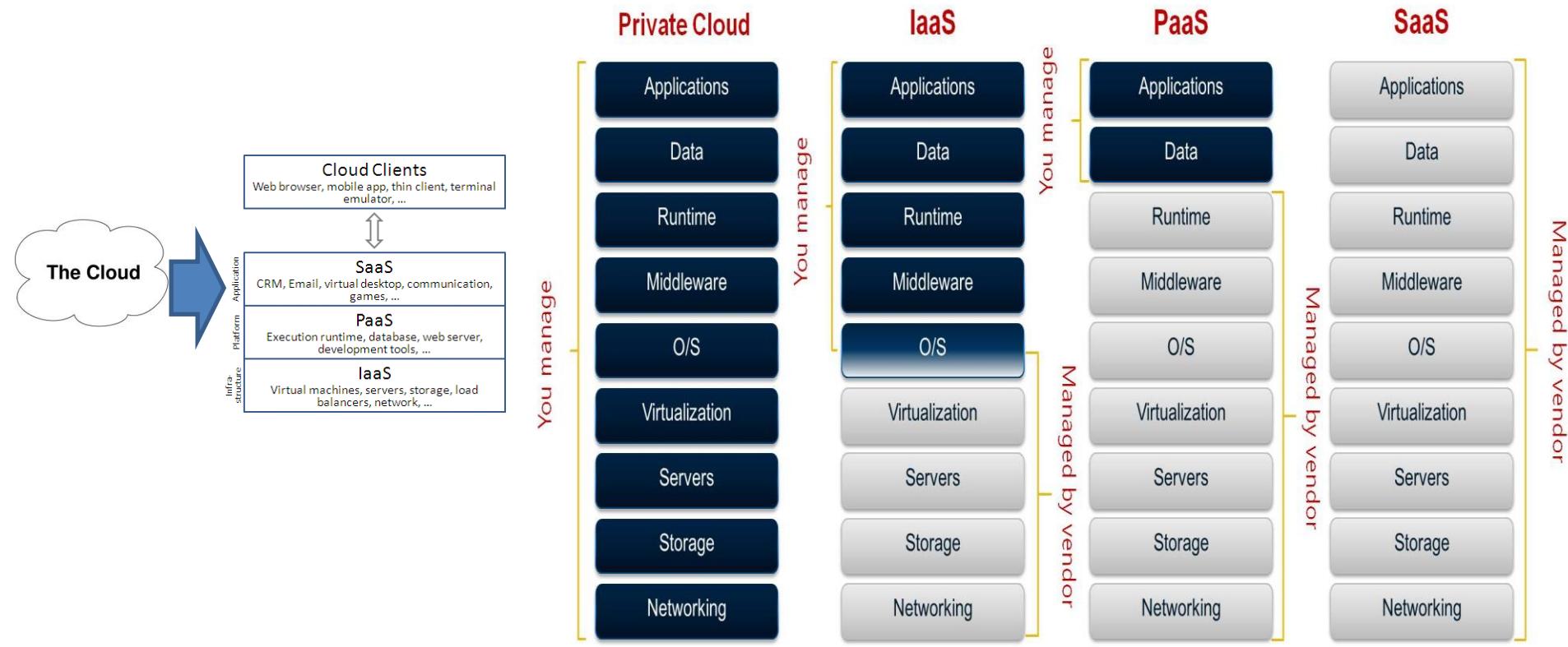


Introduction

Ok, ok got to know what to manage.....



Is it so.....????

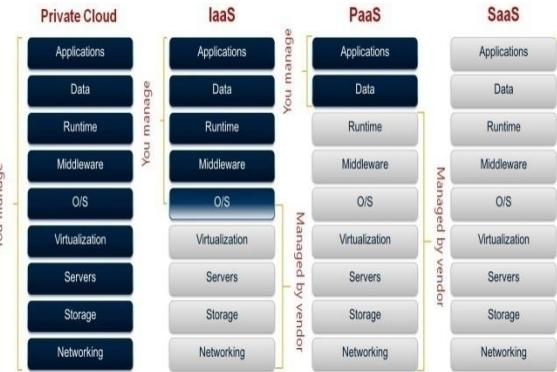




OMG!!!!

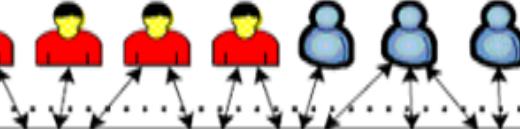


Infrastructure as a Service



**Users/
Brokers**

**SLA
Resource
Allocator**



Service Request Examiner and Admission Control
- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management

Pricing

Accounting

VM Monitor

Dispatcher

Service Request Monitor



Therefore the key requirement for cloud architecture is
“efficient management” of resources at all the three layers of cloud stack



- Cloud **Distributed environment**

- With large scale of systems to manage
- Support of multi-tenancy
- Management to maintain SLAs

- So there is need for **automation** to replace manual operations and to reduce overall cost



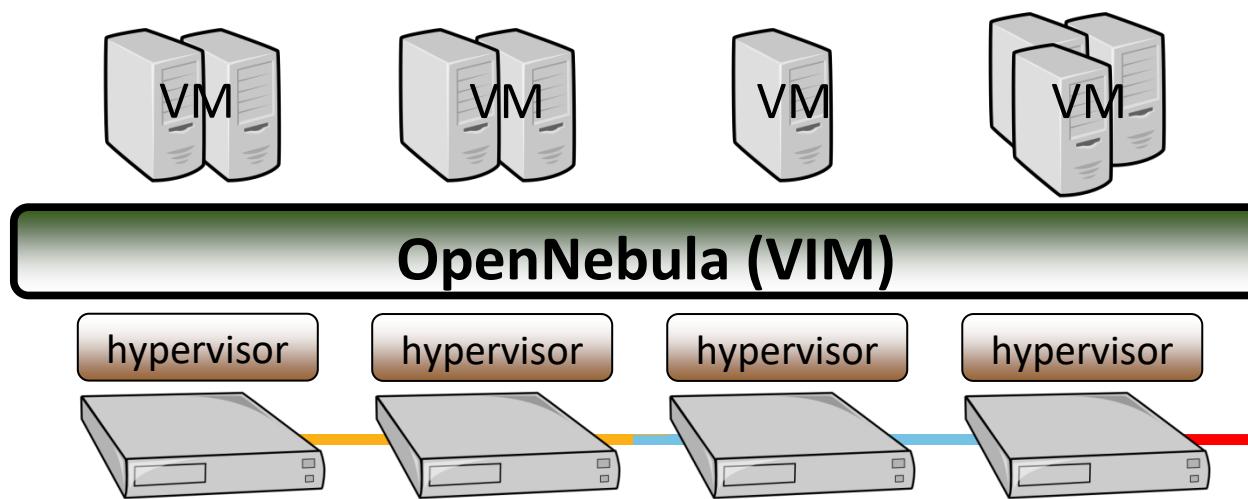
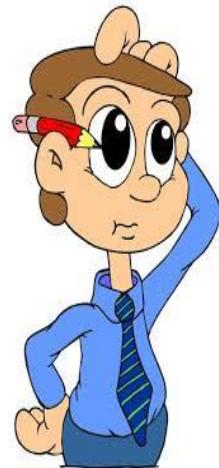


Need of the hour???

Virtual Infrastructure Managers

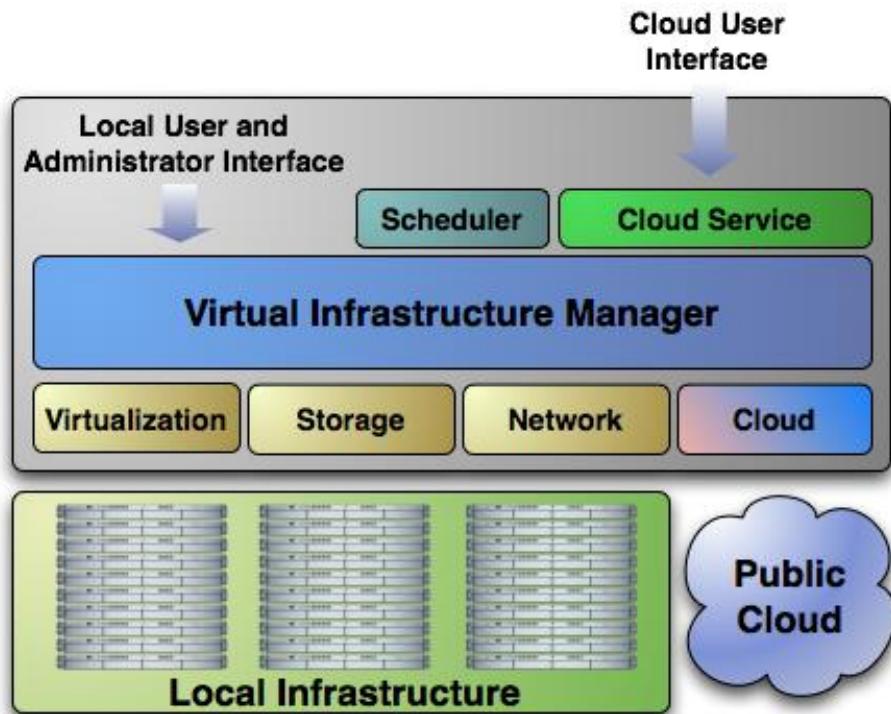
Why a Virtual Infrastructure Manager?

- VMs are great!!...but something more is needed
 - Where did/do I put my VM? (**scheduling & monitoring**)
 - How do I provision a new cluster node? (**clone**)
 - What IP addresses are available? (**networking**)
- Provide a **uniform view** of the resource pool
- **Life-cycle management** and monitoring of VM
- The VIM should **integrate** Image, Network and Virtualization



Extending the Benefits of Virtualization to Clusters

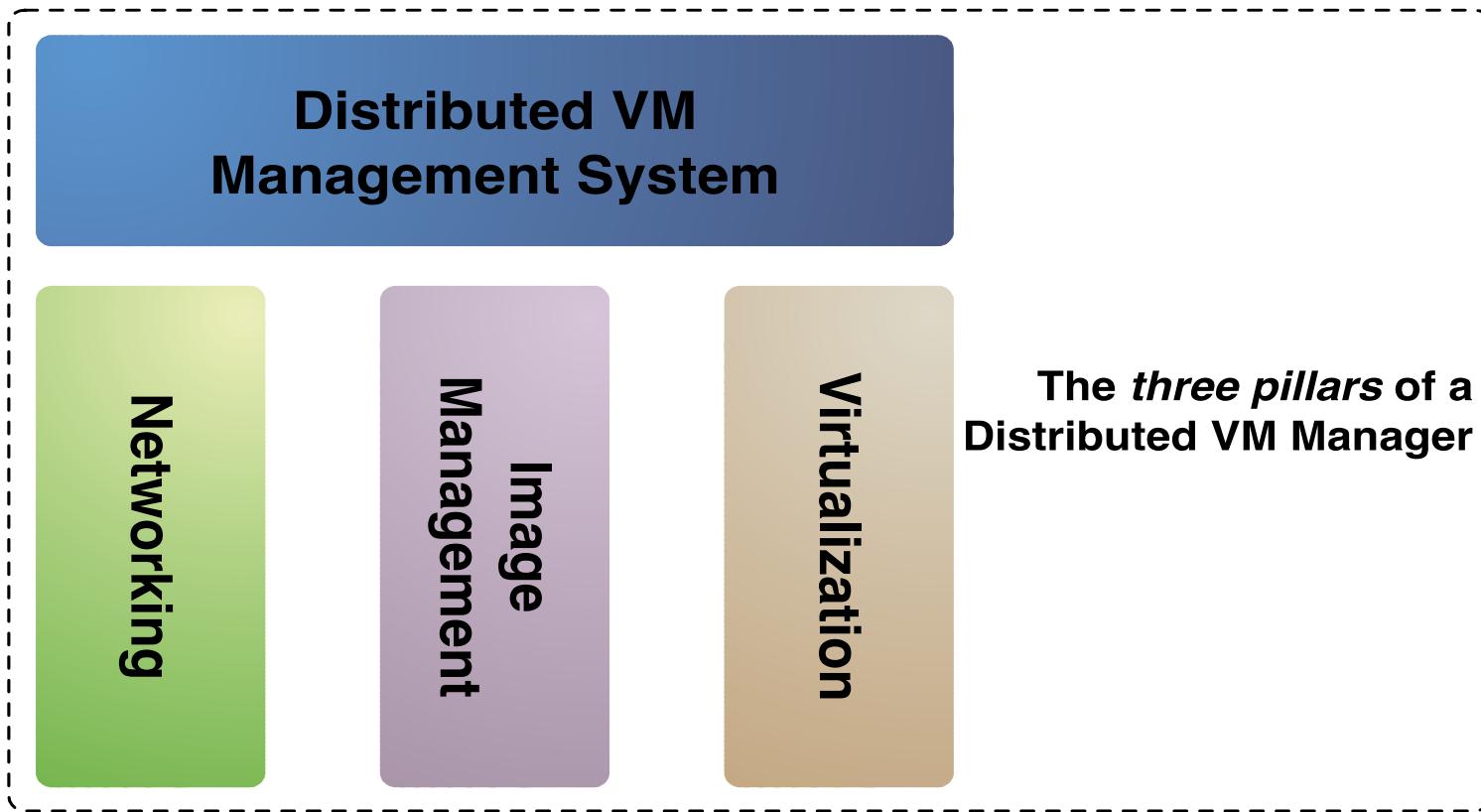
- Dynamic deployment and re-placement of virtual machines on a pool of physical resources
- Transform a rigid distributed physical infrastructure into a flexible and agile virtual infrastructure



- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

Virtual Machine Management Model

Distributed VM Management Model

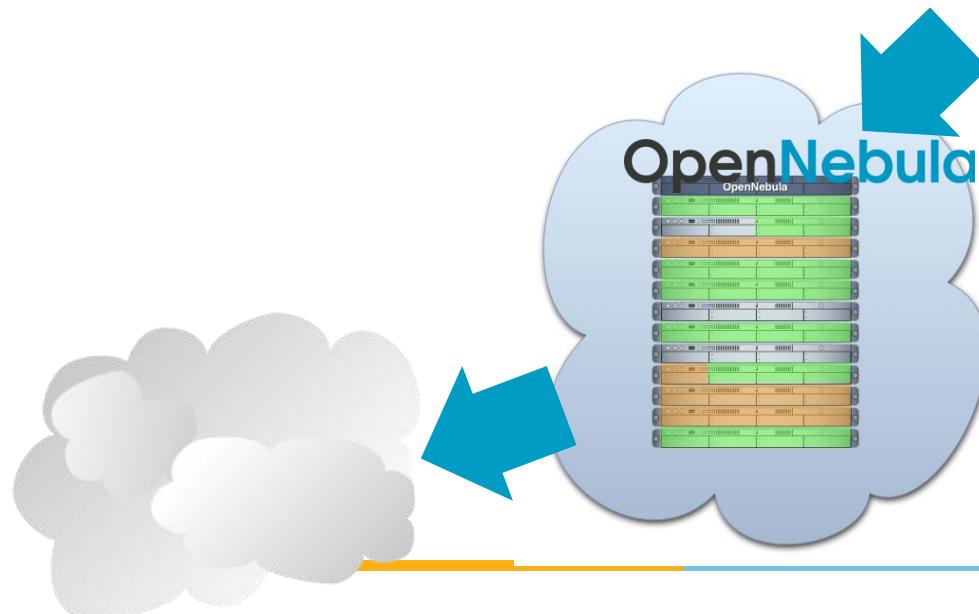


What is OpenNebula?

What is OpenNebula?

Enabling Technology to Build your Cloud

- Private Cloud to simplify and optimize internal operations
- Hybrid Cloud to supplement the capacity of the Private Cloud
- Public Cloud to expose your Private to external users



The Benefits of OpenNebula

For the Infrastructure Manager

- Centralized management of VM workload and distributed infrastructures
- Support for VM placement policies: balance of workload, server consolidation...
- Dynamic resizing of the infrastructure
- Dynamic partition and isolation of clusters
- Dynamic scaling of private infrastructure to meet fluctuating demands
- Lower infrastructure expenses combining local and remote Cloud resources

For the Infrastructure User

- Faster delivery and scalability of services
- Support for heterogeneous execution environments
- Full control of the lifecycle of virtualized services management

Interoperability from the Cloud Provider perspective

Interoperable (platform independent), innovative (feature-rich) and proven (mature to run in production).

Virtualization Drivers



Configuration



Storage

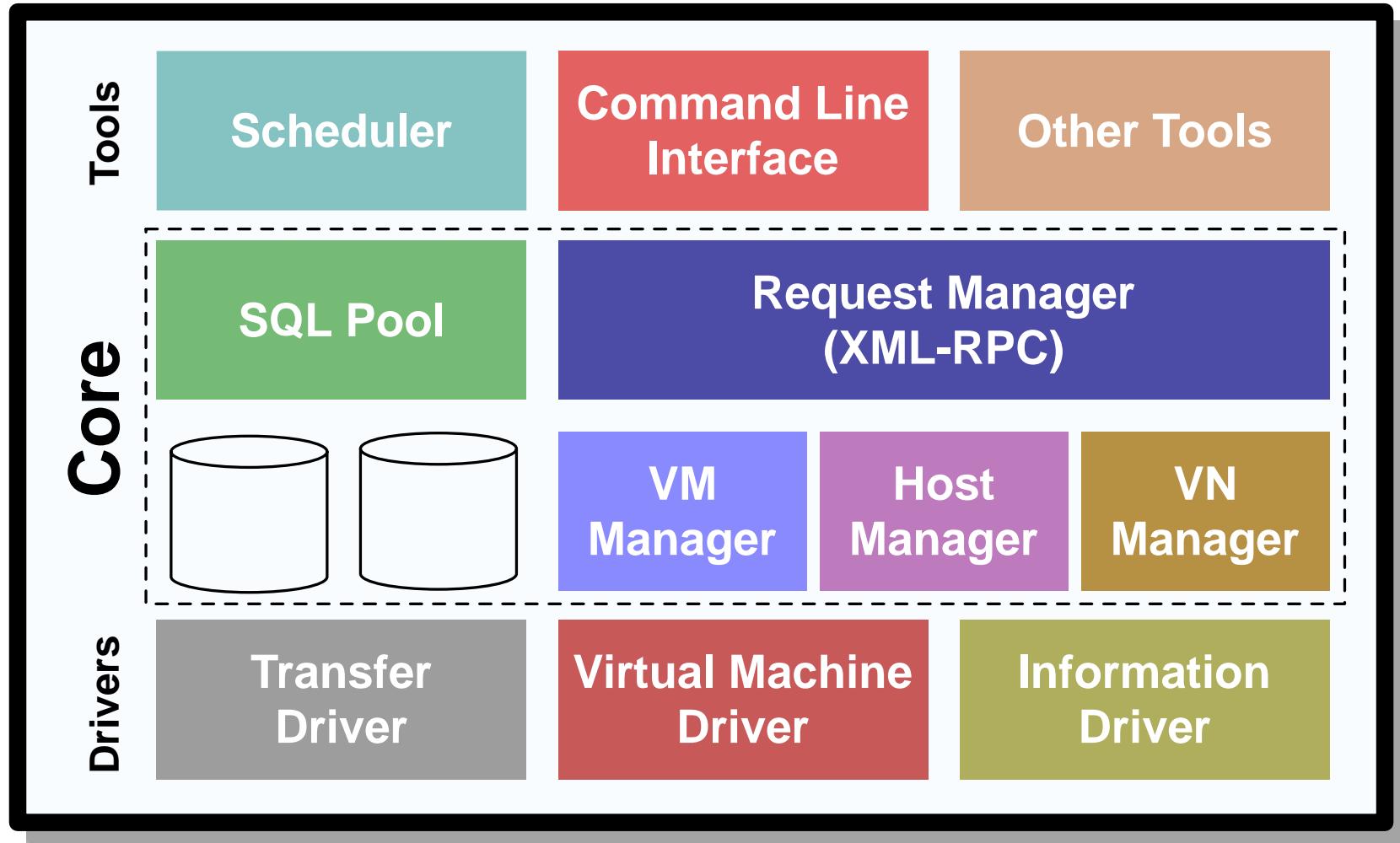


The main features of OpenNebula

Feature	Function
Internal Interface	<ul style="list-style-type: none">Unix-like CLI for fully management of VM life-cycle and physical boxesXML-RPC API and libvirt virtualization API
Scheduler	<ul style="list-style-type: none">Requirement/rank matchmaker allowing the definition of workload and resource-aware allocation policiesSupport for advance reservation of capacity through Haizea
Virtualization Management	<ul style="list-style-type: none">Xen, KVM, and VMwareGeneric libvirt connector (VirtualBox planned for 1.4.2)
Image Management	<ul style="list-style-type: none">General mechanisms to transfer and clone VM images
Network Management	<ul style="list-style-type: none">Definition of isolated virtual networks to interconnect VMs
Service Management and Contextualization	<ul style="list-style-type: none">Support for multi-tier services consisting of groups of inter-connected VMs, and their auto-configuration at boot time
Security	<ul style="list-style-type: none">Management of users by the infrastructure administrator
Fault Tolerance	<ul style="list-style-type: none">Persistent database backend to store host and VM information
Scalability	<ul style="list-style-type: none">Tested in the management of medium scale infrastructures with hundreds of servers and VMs (no scalability issues has been reported)
Flexibility and Extensibility	<ul style="list-style-type: none">Open, flexible and extensible architecture, interfaces and components, allowing its integration with any product or tool

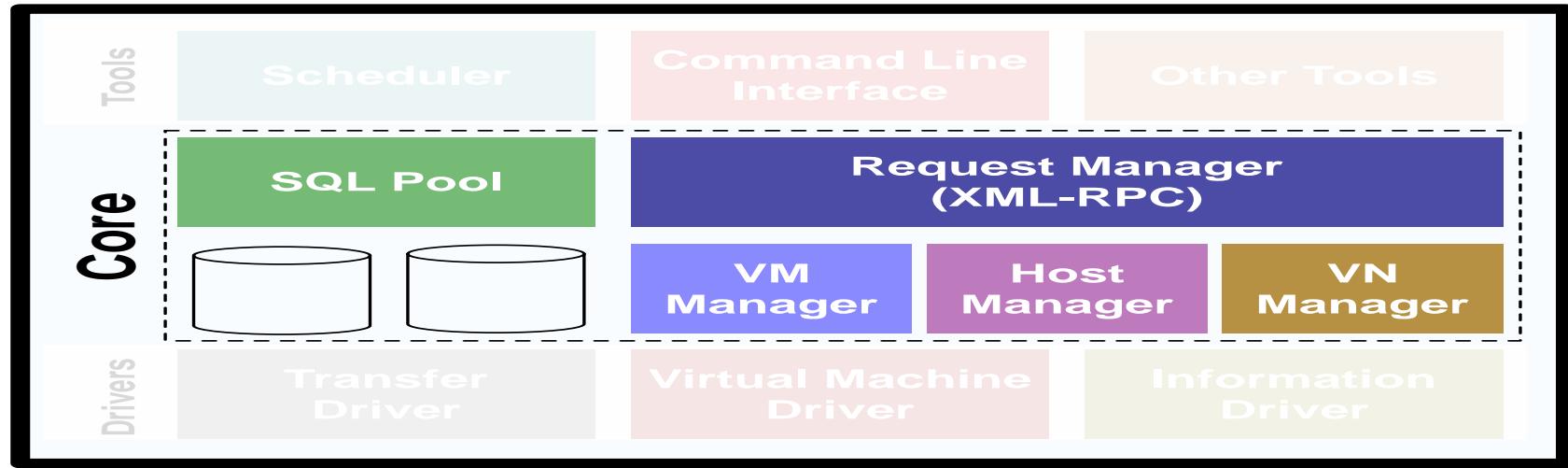
Inside OpenNebula

OpenNebula Architecture



The Core

- Request manager: Provides a XML-RPC interface to manage and get information about ONE entities.
- SQL Pool: Database that holds the state of ONE entities.
- VM Manager (virtual machine): Takes care of the VM life cycle.
- Host Manager: Holds handling information about hosts.
- VN Manager (virtual network): This component is in charge of generating MAC and IP addresses.



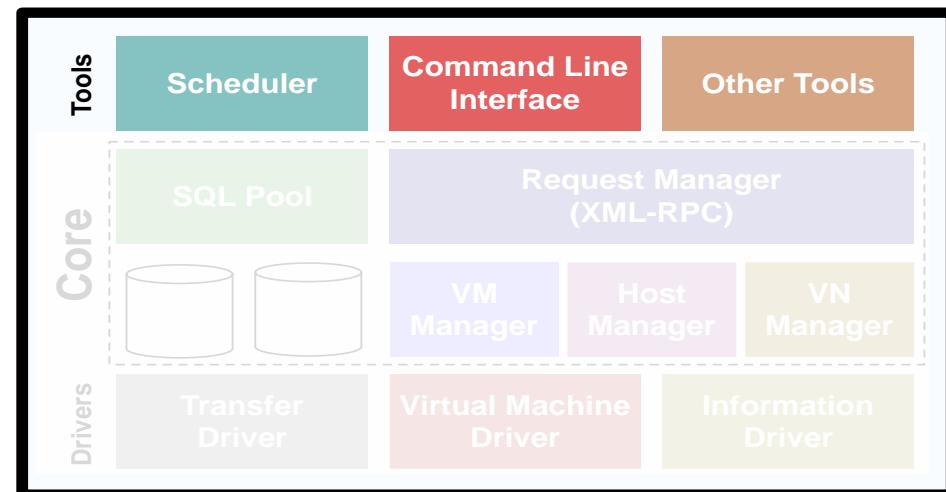
The tools layer

Scheduler:

- Searches for physical hosts to deploy newly defined VMs

Command Line Interface:

- Commands to manage OpenNebula.
- onevm: Virtual Machines
 - create, list, migrate...
- onehost: Hosts
 - create, list, disable...
- onevnet: Virtual Networks
 - create, list, delete...



The drivers layer

Transfer Driver: Takes care of the images.

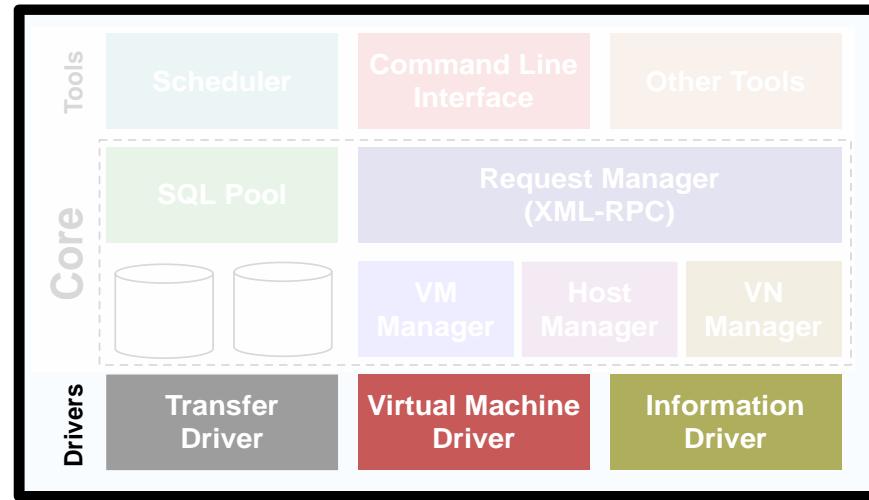
- cloning, deleting, creating swap image...

Virtual Machine Driver: Manager of the lifecycle of a virtual machine

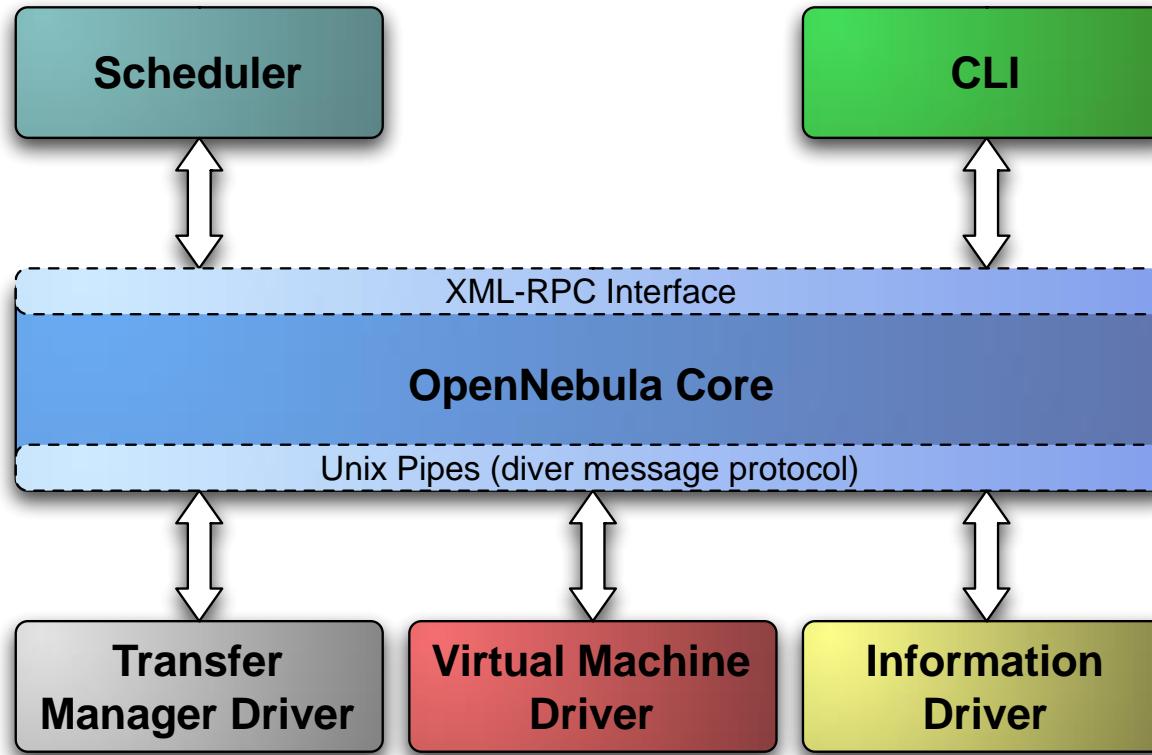
- deploy, shutdown, poll, migrate...

Information Driver: Executes scripts in physical hosts to gather information about them

- total memory, free memory, total #cpus, cpu consumed...



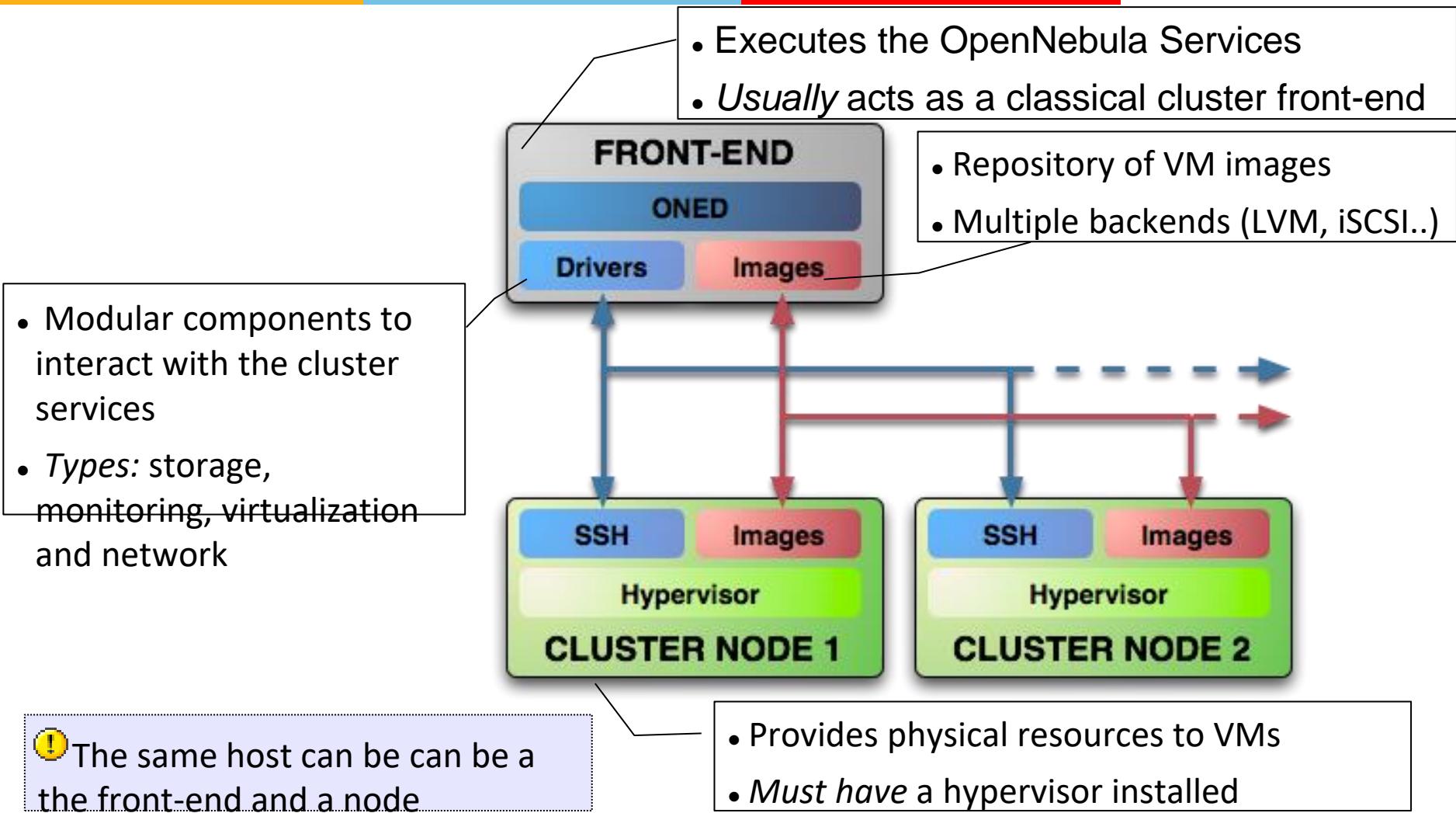
Process separation



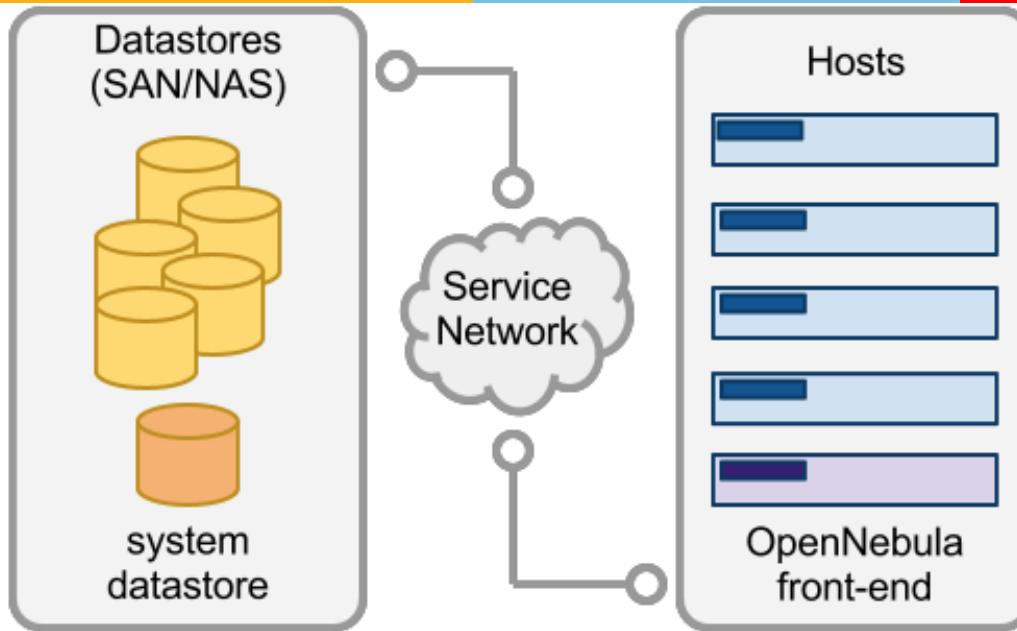
- Scheduler is a separated process, just like command line interface.
- Drivers are also separated processes using a simple text messaging protocol to communicate with OpenNebula Core Daemon (oned)

Constructing a private cloud

System Overview



Complex Storage behind OpenNebula



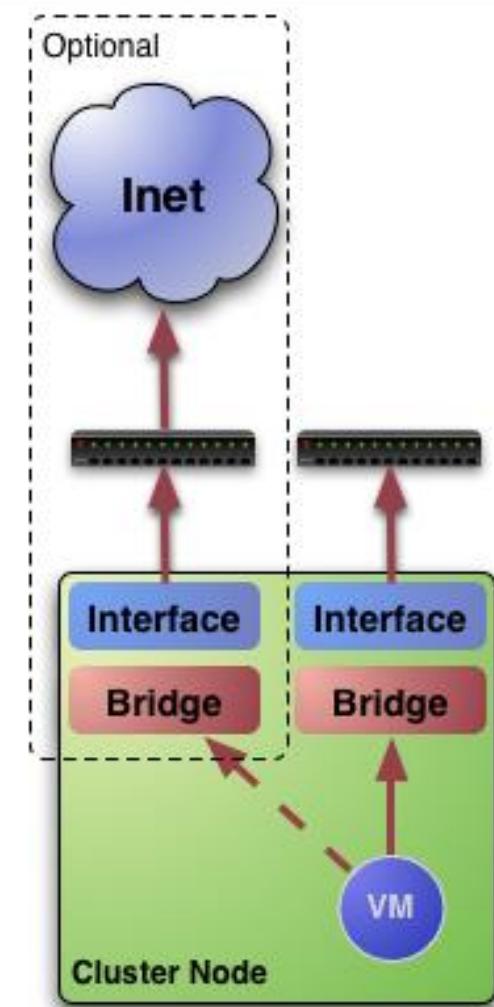
Datastore	Transfer Manager Drivers				
	shared	ssh	iscsi	qcow	vmware
System	OK	OK			
File-System	OK	OK		OK	
iSCSI			OK		
VMware	OK	OK			OK

Virtual machines and their images are represented as files

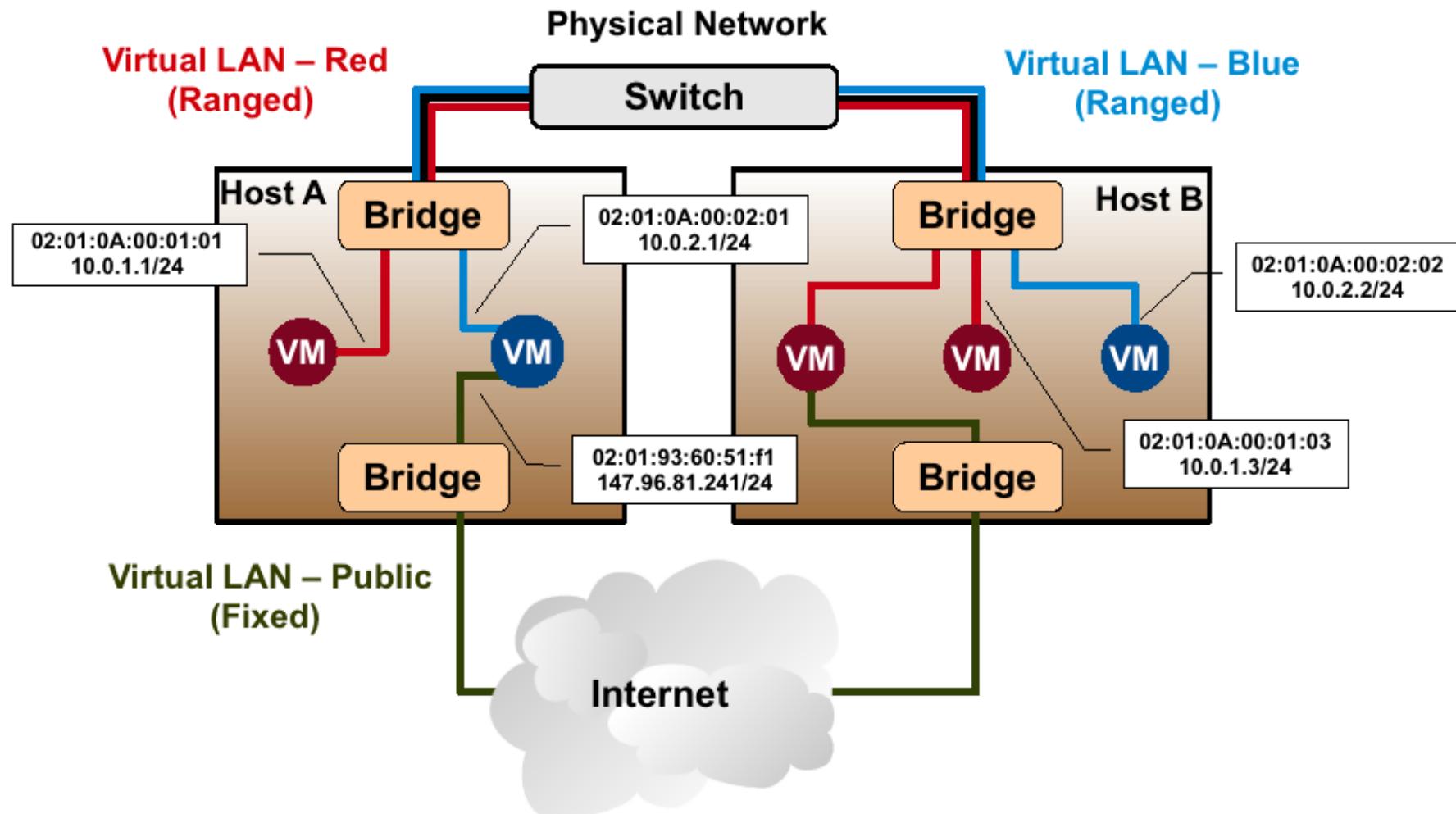
Virtual machines and their images are represented as block devices (just like a disk)

Networking for private clouds

- OpenNebula management operations use ssh connections
- ***Image traffic***, may require the movement of heavy files (VM images, checkpoints). Dedicated storage links may be a good idea
- ***VM demands***, consider the typical requirements of your VMs. Several NICs to support the VM traffic may be a good idea
- OpenNebula relies on bridge networking for the VMs



Example network setup in a private cloud



Virtual machines

VM Description

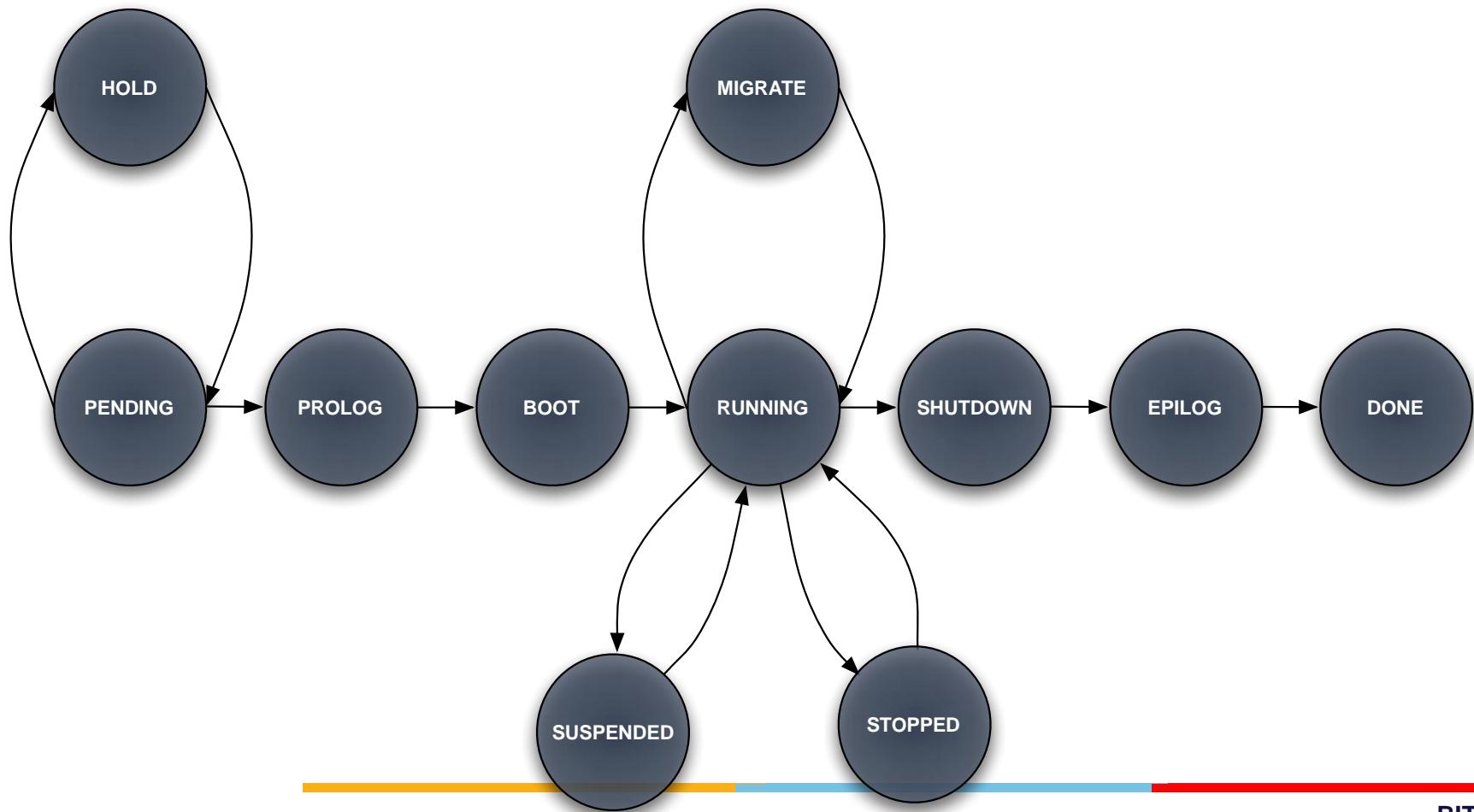
Option	Description
NAME	<ul style="list-style-type: none">Name that the VM will get for description purposes.
CPU	<ul style="list-style-type: none">Percentage of CPU divided by 100 required for the Virtual Machine.
OS (KERNEL, INITRD)	<ul style="list-style-type: none">Path of the kernel and initrd files to boot from.
DISK (SOURCE, TARGET, CLONE, TYPE)	<ul style="list-style-type: none">Description of a disk image to attach to the VM.
NIC (NETWORK)	<ul style="list-style-type: none">Definition of a virtual network the VM will be attached to.

Multiple disk and network interfaces can be specified just adding more disk/nic statements.

To create swap images you can specify TYPE=swap, SIZE=<size in MB>.

By default disk images are cloned, if you do not want that to happen CLONE=no can be specified and the VM will attach the original image.

VM States overview



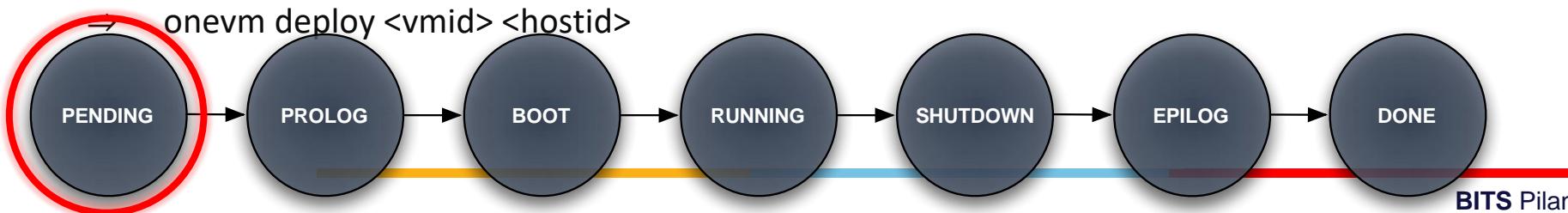
Pending state

After submitting a VM description to ONE it is added to the database and its state is set to PENDING.

In this state IP and MAC addresses are also chosen if they are not explicitly defined.

The scheduler awakes every 30 seconds and looks for VM descriptions in PENDING state and searches for a physical node that meets its requirements. Then a deploy XML-RPC message is sent to *oned* to make it run in the selected node.

Deployment can be also made manually using the Command Line Interface:

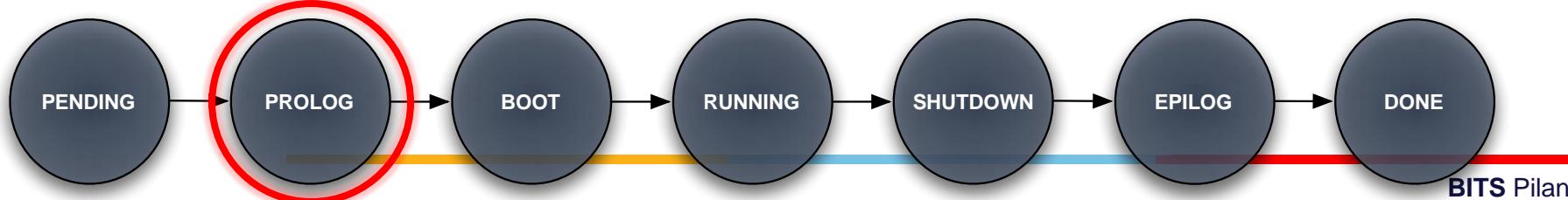


Prolog state

In PROLOG state the Transfer Driver prepares the images to be used by the VM.

Transfer actions:

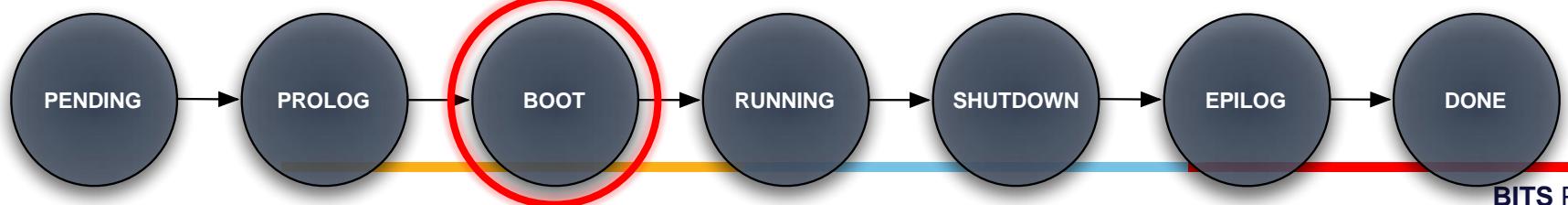
- **CLONE**: Makes a copy of a disk image file to be used by the VM. If Clone option for that file is set to false and the Transfer Driver is configured for NFS then a symbolic link is created.
- **MKSWAP**: Creates a swap disk image on the fly to be used by the VM if it is specified in the VM description.



Boot state

In this state a deployment file specific for the virtualization technology configured for the physical host is generated using the information provided in the VM description file. Then Virtual Machine Driver sends deploy command to the virtual host to start the VM.

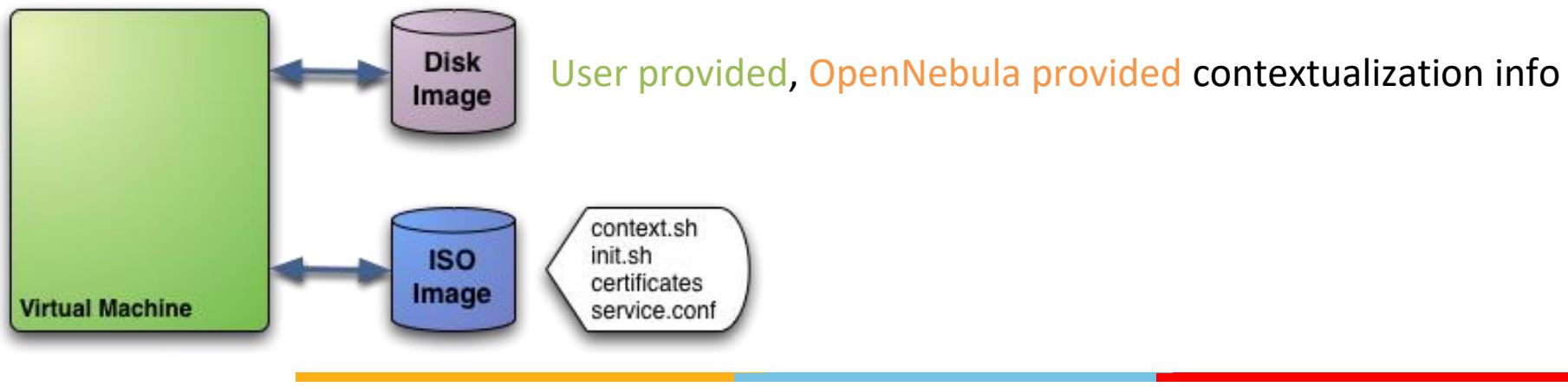
The VM will be in this state until deployment finishes or fails.



Contextualization

The ISO image has the contextualization for that VM:

- **context.sh**: contains configuration variables
- **init.sh**: script called by VM at start to configure specific services
- **certificates**: directory that contains certificates for some service
- **service.conf**: service configuration



Running and Shutdown states

While the VM is in RUNNING state it will be periodically polled to get its consumption and state.

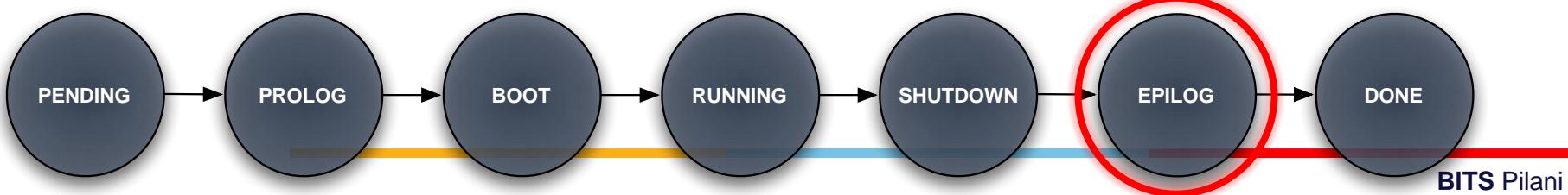
In SHUTDOWN state Virtual Machine Driver will send the shutdown command to the underlying virtual infrastructure.



Epilog state

In EPILOG state the Transfer Manager Driver is called again to perform this actions:

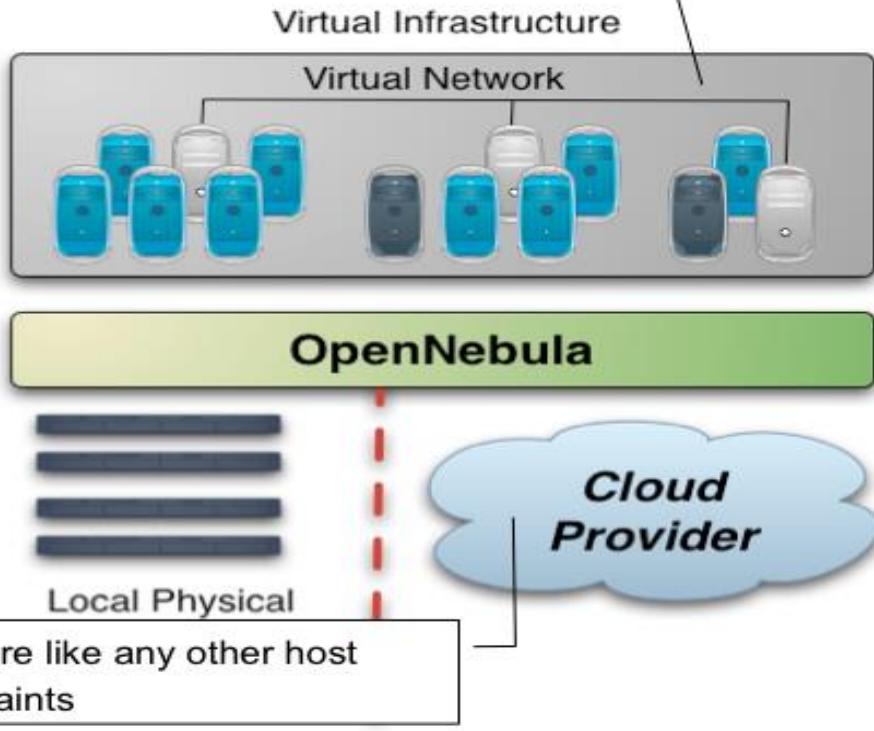
- Copy back the images that have **SAVE=yes** option.
- Delete images that were cloned or generated by **MKSWAP**.



Hybrid cloud

Overview

- VMs can be local or remote
- VM connectivity has to be configured, usually VPNs



Making an Amazon EC2 hybrid

Amazon EC2 cloud is managed by OpenNebula as any other cluster node

- You can use several accounts by adding a driver for each account (use the arguments attribute, -k and -c options). Then create a host that uses the driver
- You can use multiple EC2 zones, add a driver for each zone (use the arguments attribute, -u option), and a host that uses that driver
- You can limit the use of EC2 instances by modifying the IM file

Using an EC2 hybrid cloud

Virtual Machines can be instantiated locally or in EC2

The VM template must provide a description for both instantiation methods.

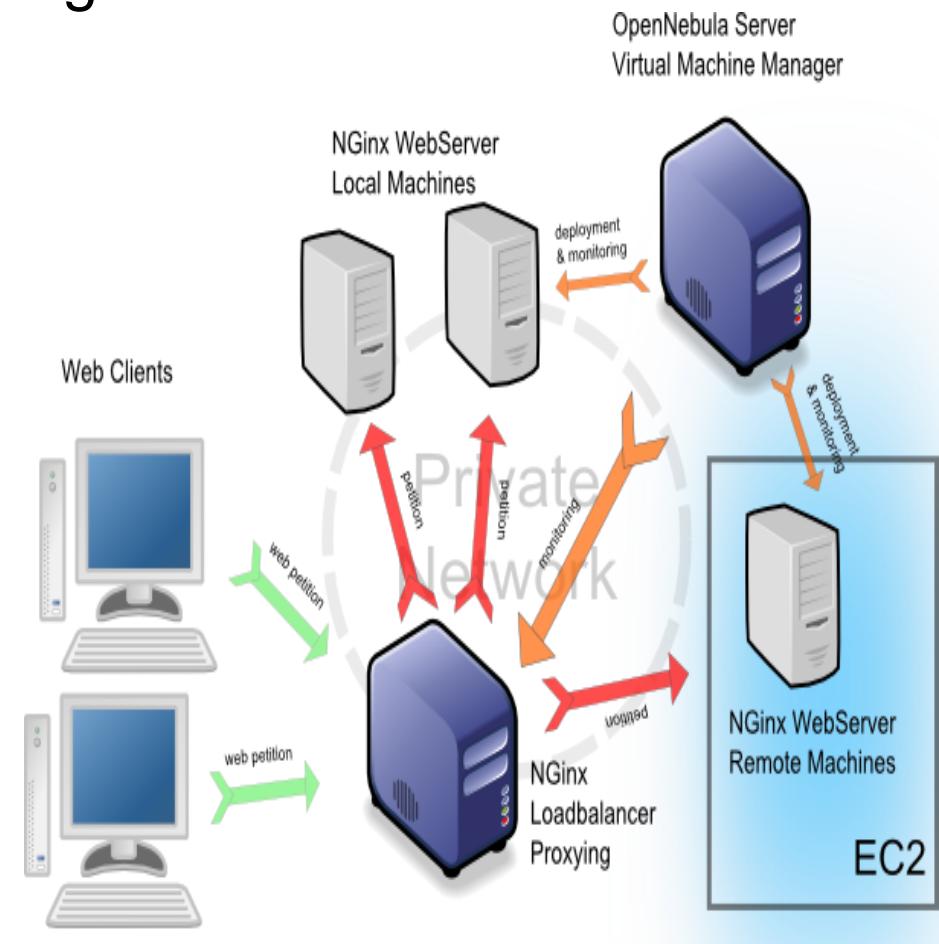
The EC2 counterpart of your VM (AMI_ID) must be available for the driver account

The EC2 VM template attribute should describe not only the VM's properties but the contact details of the external cloud provider

Hybrid cloud Use Case

On-demand Scaling of Computing Clusters

- On-demand Scaling of Web Servers
- Elastic execution of the NGinx web server
- The capacity of the elastic web application can be dynamically increased or decreased by adding or removing NGinx instances





BITS Pilani

Cloud Computing

SEWP ZG527

Agenda

- Cloud Security Issues
- Threat Models
- Goals of Cloud Computing Security.
- Availability
- Multi-Tenancy
- Service Level Agreements(SLAs)

Introduction to cloud security

If cloud computing is so great, why isn't everyone doing it?

- The cloud acts as a big black box, nothing inside the cloud is visible to the clients
- Clients have no idea or control over what happens inside a cloud
- Even if the cloud provider is honest, it can have malicious system admins who can tamper with the VMs and violate confidentiality and integrity
- Clouds are still subject to traditional data confidentiality, integrity, availability, and privacy issues, plus some additional attacks

Cloud Security Issues

- **Most security problems stem from:**
 - Loss of Control
 - Take back control
 - Data and apps may still need to be on the cloud
 - But can they be managed in some way by the consumer?
 - Lack of trust
 - Increase trust (mechanisms)
 - Technology
 - Policy, regulation
 - Contracts (incentives): topic of a future talk
 - Multi-tenancy
 - Private cloud
 - Takes away the reasons to use a cloud in the first place
 - VPC: its still not a separate system
 - Strong separation

Loss of Control in the Cloud

Consumer's loss of control

- Data, applications, resources are located with provider
- User identity management is handled by the cloud
- User access control rules, security policies and enforcement are managed by the cloud provider
- Consumer relies on provider to ensure
 - Data security and privacy
 - Resource availability
 - Monitoring and repairing of services/resources

Multi-tenancy Issues in the Cloud

- Conflict between tenants' opposing goals
 - Tenants share a pool of resources and have opposing goals
 - How does multi-tenancy deal with conflict of interest?
 - Can tenants get along together and 'play nicely' ?
 - If they can't, can we isolate them?
 - How to provide separation between tenants?
 - Cloud Computing brings new threats

Multiple independent users share the same physical infrastructure

Thus an attacker can legitimately be in the same physical machine as the target

Taxonomy of Fear

- **Confidentiality**
 - Fear of loss of control over data
 - Will the sensitive data stored on a cloud remain confidential?
 - Will cloud compromises leak confidential client data
 - Will the cloud provider itself be honest and won't peek into the data?
- **Integrity**
 - How do I know that the cloud provider is doing the computations correctly?
 - How do I ensure that the cloud provider really stored my data without tampering with it?

Taxonomy of Fear

Availability

- Will critical systems go down at the client, if the provider is attacked in a Denial of Service attack?
- What happens if cloud provider goes out of business?
- Would cloud scale well-enough?
- Although cloud providers argue their downtime compares well with cloud user's own data centers

Taxonomy of Fear (Cont...)

- Privacy issues raised via massive data mining
 - Cloud now stores data from a lot of clients, and can run data mining algorithms to get large amounts of information on clients
- Increased attack surface
 - Entity outside the organization now stores and computes data, and so
 - Attackers can now target the communication link between cloud provider and client
 - Cloud provider employees can be phished

Taxonomy of Fear (Cont...)

- Auditability and forensics (out of control of data)
 - Difficult to audit data held outside organisation in a cloud
 - Forensics also made difficult since now clients don't maintain data locally
- Legal quagmire and transitive trust issues
 - Who is responsible for complying with regulations?. CSA (Cloud Security Alliance), etc.
 - If cloud provider subcontracts to third party clouds, will the data still be secure?

Threat Model

- Basic components
 - Attacker modelling
 - Choose what attacker to consider
 - insider vs. **outsider**?
 - single vs. collaborator?
 - Attacker motivation and capabilities
 - Attacker goals

Threat Model (Cont...)

- A threat model helps in analyzing a security problem, design mitigation strategies, and evaluate solutions
- Steps:
 - Identify attackers, assets, threats and other components
 - Rank the threats
 - Choose mitigation strategies
 - Build solutions based on the strategies

Cloud Information Security Objectives

Cloud security has unique challenges due to:

- Shared infrastructure
- Rapid movement of servers and workload in the infrastructure

The **basic objectives** of cloud security are:

- 1) **Confidentiality:** Prevention of intentional or unintentional unauthorized disclosure of information
- 2) **Integrity:** Modifications are not made to data by unauthorized personnel or processes
- 3) **Availability:** To ensure the **reliable and timely access to cloud data** or cloud computing resources by the appropriate personnel

Additional Objectives of Cloud Security

These include:

- i. **Cost-effectiveness:** Security implementation should not greatly increase the cost of a cloud solution.
- ii. **Reliability and performance:** These should not be greatly impacted by cloud security.

Cloud Security Services

Factors that directly affect **cloud software assurance** include are:

1. Authentication:

- It is about testing the user's identity and ensures that users are who they claim to be.
- For example, a user presents an identity (user ID) to a computer login screen and then has to provide a password. **The computer system authenticates the user by verifying that the password corresponds to the individual presenting the ID.**

Cloud Security Services (Cont...)

2) Authorization:

- It refers to rights and privileges granted to an individual or process that enable **access to computer resources and information assets**.
- Once a user's identity and authentication are established, authorization levels determine the extent of system rights a user can hold.

3) Auditing:

- To maintain operational assurance, **organizations use two basic methods: System audits and Monitoring**.
- A ***system audit*** is a one-time or periodic event to evaluate security.
- ***Monitoring*** refers to an ongoing activity that examines either the system or the users, such as **intrusion detection**.

Cloud Security Services (Cont...)

3) Auditing: (Cont...)

IT auditors typically audit the following functions:

- System and transaction controls
- Systems development standards
- Backup controls
- Data center security, etc.

Cloud Security Services (Cont...)

Audit Log:

An *audit trail or log* is a set of records that collectively provide **documentary evidence of processing, used to aid in tracing from original transactions forward to related records and reports, and/or backward from records and reports to their component source transactions.**

Audit logs should record the following:

- The transaction's date and time
- Who processed the transaction
- At which terminal the transaction was processed
- Various security events relating to the transaction

Cloud Security Services (Cont...)

4. Accountability: It is the ability to **determine the actions and behaviors of a single individual** within a cloud system and **to identify that particular individual.**

- Audit trails and logs support accountability and can be used to conduct postmortem studies in order to analyze historical events and the individuals or processes associated with those events.

Note: Accountability is related to the concept of *nonrepudiation*, wherein an individual cannot successfully deny the performance of an action.

Threats and Vulnerabilities

Def (Threat):

- It is simply any event that, if realized, can cause damage to a system and create a loss of confidentiality, availability, or integrity.
- Threats can be malicious, such as the intentional modification of sensitive information, or they can be accidental — such as an error in a transaction calculation or the accidental deletion of a file

Def(Vulnerability):

- A *vulnerability* is a weakness in a system that can be exploited by a threat.
- Reducing the vulnerable aspects of a system can reduce the risk and impact of threats on the system. For example, a password-generation tool, which helps users choose robust passwords, reduces the chance that users will select poor passwords (the vulnerability) and makes the password more difficult to crack (the threat of external attack).

Common Threats to both Cloud and Traditional Infrastructure

1. **Eavesdropping:** Data scavenging, traffic or trend analysis, social engineering, economic or political espionage, sniffing, dumpster diving, keystroke monitoring are all types of eavesdropping to gain information or to create a foundation for a later attack.
Note: Eavesdropping is a primary cause of the failure of confidentiality.
2. **Fraud** — Examples of fraud include collusion, falsified transactions, data manipulation, and other altering of data integrity for gain.
3. **Theft** — Examples of theft include the theft of information or trade secrets for profit or unauthorized disclosure, and physical theft of hardware or software.
4. **Sabotage** — Sabotage includes denial-of-service (DoS) attacks, production delays, and data integrity sabotage.
5. **External attack** — Examples of external attacks include malicious cracking, scanning, and probing to gain infrastructure information, demon dialing to locate an unsecured modem line, and the insertion of a malicious code or virus.

Common types of attacks

1. **Logon Abuse:** It refers to legitimate users accessing services of a higher security level that would normally be restricted to them.
2. ***Inappropriate System Use:*** It refers to the nonbusiness **or personal use of a network** by otherwise authorized users, such as Internet surfing to **inappropriate content sites** (travel, pornography, sports, and so forth).
3. ***Eavesdropping:*** It is a type of network attack that consists of the unauthorized interception of network traffic.

Common types of attacks (Cont...)

4. Network Intrusion: It refers to the use of unauthorized access to break into a network primarily from an external source. The attackers exploit known security vulnerabilities in the security perimeter.

5. Denial-of-Service (DoS) Attacks:

- It occurs when legitimate users are unable to access information systems, devices, or other network resources due to the actions of a malicious attacker.
- It is accomplished by flooding the targeted host or network with traffic until the target cannot respond or simply crashes, preventing access for legitimate users

6. Session Hijacking Attacks: It refers to unauthorized access to a system. An attacker hijacks a session between a trusted client and network server.

Cloud Infrastructure Security

- The cloud consists of a **shared infrastructure** that can be rapidly configured on demand to meet business needs.
- The cloud infrastructure can be partitioned into:
 - 1) **Physical infrastructure**
 - 2) **Virtual infrastructure.**

Based on the above, the security of the cloud infrastructure can be classified as **Physical Security** and **Virtual Security**.

Physical Security

- It implies that the data center the cloud is hosted in should be secure against **physical threats**.
- It includes **attempts at penetration** by intruders, but also **protection against natural hazards** and **disasters** such as floods, and **human error** such as switching off the air conditioning.

Physical Security (Cont...)

Multi-layered system is required to ensure **physical security**. This includes:

- i. A **central monitoring and control center** with dedicated staff
- ii. Monitoring for each possible physical threat, such as intrusion, or natural hazards such as floods.
- iii. Training of the staff in response to threat situations
- iv. Manual or automated back-up systems to help contain threats (e.g., pumps to help contain the damage from floods)
- v. Secure access to the facility. This requires that the various threats to the data center be identified, and appropriate procedures derived for handling these threats.

Virtual Security

i) Cloud Time Service:

- If all systems in the datacenter are synchronized to the same clock, this is helpful both to ensure correct operation of the systems, as well as to facilitate later analysis of system logs. A common way to do this is by use of the Network Time Protocol (NTP).

2) Identity Management:

- It is a foundation for achieving confidentiality, integrity and availability.

Virtual Security (Cont...)

3. Access Management:

- It is to allow accesses to cloud facilities only to authorized users.
- Not allow unrestricted access to cloud management personnel
- Allow implementation of **multi-factor authentication** (e.g., use of a password together with a digital key) for very sensitive operations.

4. Key Management:

- In a cloud, with shared storage, encryption is a key technology to ensure isolation of access.
- Need to provide secure facilities for the **generation, assignment, revocation, and archiving of keys**. It is also necessary to generate procedures for recovering from compromised keys.

Virtual Security (Cont...)

5. Auditing:

- Auditing is needed for all system and network components.
- The audit should capture all security-related events, together with data needed to analyze the event such as the **time, system on which the event occurred, and userid** that initiated the event.
- The audit log should be centrally maintained and secure.

Virtual Security (Cont...)

6. Security Monitoring:

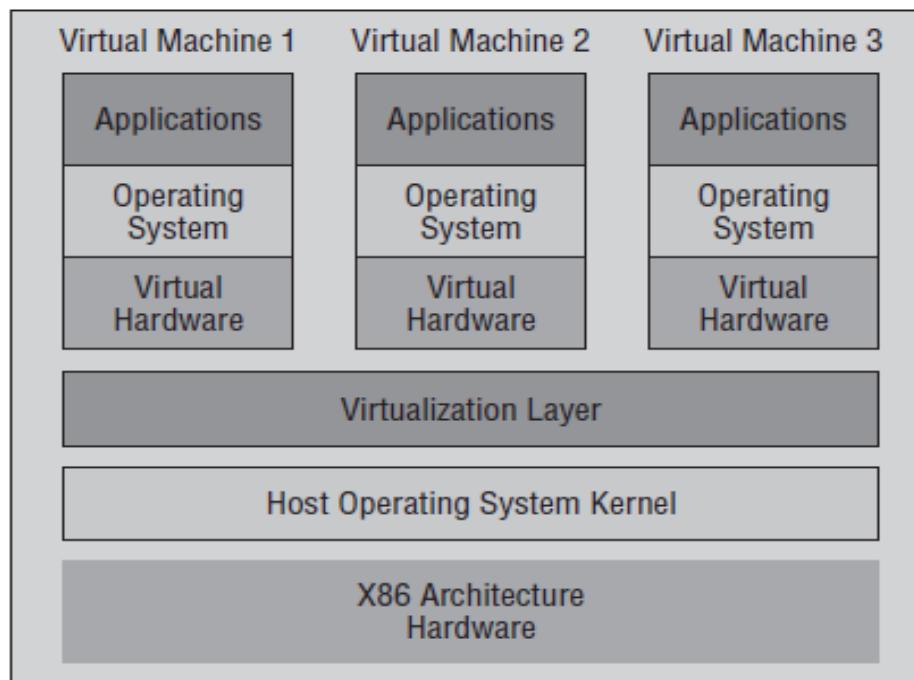
-This includes an infrastructure to **generate alerts** when a critical security event has occurred, including a cloud-wide intrusion and anomaly detection system.

7. Security Testing:

- Test all software for security before deployment in an isolated test bed.
- Patches to software should also be tested in this environment before being released into production.

Virtual Threats

- Many VM vulnerabilities stem from the fact that a vulnerability in one VM system can be exploited to attack other VM systems or the host systems, as multiple virtual machines share the same physical hardware.



Type 2 Virtualized Environment

Virtual Threats (Cont...)

- 1. Shared clipboard** — It allows data to be transferred between VMs and the host, providing a means of moving data between malicious programs in VMs of different security realms.
- 2. Keystroke logging** — Some VM technologies enable the logging of keystrokes and screen updates to be passed across virtual terminals in the virtual machine, writing to host files and permitting the monitoring of encrypted terminal connections inside the VM.

Virtual Threats (Cont...)

3. VM monitoring from the host: Because all network packets coming from or going to a VM pass through the host, the host may be able to affect the VM by various ways such as Monitoring and configuring resources available to the VMs, including CPU, memory, disk, and network usage of VMs.

Virtual Threats (Cont...)

4. **Virtual machine monitoring from another VM** — Suppose if the VM platform uses a virtual hub or switch to connect the VMs to the host, then intruders may be able to use a hacker technique known as “ARP poisoning” to redirect packets going to or from the other VM for sniffing.
5. **Virtual machine backdoors** — A backdoor, covert communications channel between the guest and host could allow intruders to perform potentially dangerous operations.

Hypervisor Risks

- The *hypervisor* is the part of a virtual machine that allows host resource sharing and enables VM/host isolation.
- The ability of the hypervisor to provide the necessary isolation during intentional attack greatly determines how well the virtual machine can survive risk.

Why the hypervisor is susceptible to risk?

Because it's a software program; risk increases as the volume and complexity of application code increases.

VM Security Recommendations

Best Practice Security Techniques: These areas include physical security, patching, and remote management techniques.

i) **Hardening the Host Operating System:** Vulnerabilities inherent in the operating system of the host computer can flow upward into the virtual machine operating system.

- A compromise of the underlying host OS would give an intruder access to all services on all virtual machines hosted by the machine.

Techniques to maintain the security posture of the underlying technology:

- Use **strong passwords**, such as lengthy, hard to guess passwords with letters, numbers, and symbol combinations, and change them often.
- Disable unneeded services or programs, especially networked services.
- Require **full authentication** for access control.
- The host should be individually **firewalled**.
- Patch and update the host regularly, after testing on a nonproduction unit.

- **Limiting Physical Access to the Host:** It is required to prevent intruders from attacking the hardware of the virtual machine. When attackers can access a host they can do the following:
- Use OS-specific keystrokes to kill processes, monitor resource usage, or shut down the machine, commonly without needing a valid login account and password

Identity Management

Note: Identification and authentication are the keystones of most access control systems.

Def(Identification):

- It is the act of a user professing an identity to a system, usually in the form of a username or user logon ID to the system.
- It establishes user accountability for the actions on the system.
- User IDs should be unique and not shared among different individuals.
- User IDs follow set standards, such as first initial followed by last name, and so on.
- An ID should not reflect the user's job title or function in order to enhance security and reduce the amount of information available to an attacker.

Identity Management

Def(Authentication):

- It is verification that the user's claimed identity is valid, and it is usually implemented through a user password at logon.

Authentication is based on the following three factor types:

- Type 1** — Something you know, such as a personal identification number (PIN) or password
- Type 2** — Something you have, such as an ATM card or smart card
- Type 3** — Something you are (physically), such as a fingerprint or retina scan

Two Factor Authentication

- It requires two of the three factors to be used in the authentication process.
- For example, withdrawing funds from an ATM machine requires two-factor authentication in the form of the ATM card (something you have) and a PIN number (something you know).

Password

- Passwords must be protected.
- In the ideal case, a password should be used only once. This “**one-time password**,” or **OTP**, provides maximum security because a new password is required for each new logon.
- A password that is the same for each logon is called a ***static password***.
- A password that changes with each logon is termed a ***dynamic password***.
- A front-end authentication device or a back-end authentication server, which services multiple workstations or the host can perform the authentication.
- Passwords can be provided by a number of devices, including **tokens, memorycards, and smart cards**.

Note: Passwords can be required to change monthly, quarterly, or at other intervals, depending on the criticality of the information needing protection and the password's frequency of use.

Note: *Tokens*, in the form of small, hand-held **devices**, are used to provide passwords.

Biometrics

- It is an alternative to using passwords for authentication.
- Biometrics is based on the Type 3 authentication mechanism.
- Biometrics is defined as an automated means of identifying or authenticating the identity of a living person based on physiological or behavioral characteristics.
- Identification in biometrics is a **one-to-many search** of an individual's characteristics from a database of stored images while Authentication is a **one-to-one search** to verify a claim to an identity made by a person.
- Biometrics is used for identification in physical controls and for authentication in logical controls.

Biometrics (Cont...)

Three main performance measures in biometrics

- 1. False rejection rate (FRR) or Type I Error :** The percentage of valid subjects that are falsely rejected.
- 2. False acceptance rate (FAR) or Type II Error :** The percentage of invalid subjects that are falsely accepted.
- 3. Crossover error rate (CER) :** The percentage at which the FRR equals the FAR. The smaller the CER, the better the device is performing.

Biometrics (Cont...)

Typical biometric characteristics that are used to uniquely authenticate an individual's identity:

- **Fingerprints** — Fingerprint characteristics are captured and stored. Typical CERs are 4–5%.
- **Retina scans** — The eye is placed approximately two inches from a camera and an invisible light source scans the retina for blood vessel patterns. CERs are approximately 1.4%.
- **Iris scans** — A video camera remotely captures iris patterns and characteristics. CER values are around 0.5%.
- **Hand geometry** — Cameras capture three-dimensional hand characteristics. CERs are approximately 2%.
- **Voice** — Sensors capture voice characteristics, including throat vibrations and air pressure, when the subject speaks a phrase. CERs are in the range of 10%.

Note: Other types of biometric characteristics include facial and palm scans.

Implementing Identity Management

- Establishing a database of identities and credentials
- Managing users' access rights
- Enforcing security policy
- Developing the capability to create and modify accounts
- Setting up monitoring of resource accesses
- Installing a procedure for removing access rights
- Providing training in proper procedures

Access Control

- Access control is intrinsically tied to identity management and is necessary to preserve the **confidentiality, integrity, and availability of cloud data.**
- Three things that must be considered for the implementation of access control mechanisms: **Threats** to the system, the **system's vulnerability** to these threats, and the **risk** (potential for harm or loss to an information system or network) that the threats might materialize.
- Benefits of access control: Keep Track of Employees, Reduce Theft and Accidents, Secure Sensitive Documents and Data, etc.

Controls

- Controls are implemented to mitigate risk and reduce the potential for loss.
- *Two important concepts of Controls: **Separation of duties** and the **principle of least privilege**.*

Def(Separation of duties): It requires an activity or process to be performed by two or more entities for successful completion. Thus, the only way that a security policy can be violated is if there is collusion among the entities.

Any example could you provide?.

- For example, in a financial environment, the person requesting that a check be issued for payment should not also be the person who has authority to sign the check.

Def(Least privilege): The minimum resources and privileges required to complete the task for the minimum necessary period of time.

Control Measures

Control measures can be administrative, logical (also called technical), and physical in their implementation. They are:

- **Administrative controls** include **policies and procedures**, security awareness training, background checks, work habit checks, a review of vacation history, and increased supervision.
- **Logical or technical controls** involve the **restriction of access to systems and the protection of information**. Examples of these types of controls are encryption, smart cards, access control lists, and transmission protocols.
- **Physical controls** incorporate **guards and building security** in general, such as the locking of doors, the securing of server rooms or laptops, the protection of cables, the separation of duties, and the backing up of files.

Models for Controlling Access

Controlling access by a **subject** (an active entity such as an individual or process) to an **object** (a passive entity such as a file) involves setting up access rules. These rules can be classified into three categories or models. They are:

- 1. Mandatory Access Control**
- 2. Discretionary Access Control**
- 3. Nondiscretionary Access Control**

1. Mandatory Access Control

- The authorization of a subject's access to an object depends **upon labels**, which indicate the **subject's clearance, and the classification or sensitivity** of the object.
- For example, the military classifies documents as unclassified, confidential, secret, and top secret.
- Similarly, an individual can receive a clearance of confidential, secret, or top secret and can have access to **documents classified at or below his or her specified clearance level**. Thus, an individual with a clearance of “secret” can have access to secret and confidential documents with a restriction.
- **Rule-based access control** is a type of mandatory access control because rules determine this access (such as the correspondence of clearance labels to classification labels), rather than the identity of the subjects and objects alone.

2. Discretionary Access Control

- It allows to decide on who can access which areas of the premises or resources.
- In this model, the **subject has authority**, within certain limitations, to specify what objects are accessible.
- For example, **access control lists (ACLs)** can be used.
- An **access control list** is a list denoting which users have what privileges to a particular resource.
- For example, a *tabular listing* would show the subjects or users who have access to the object, e.g., file X, and what privileges they have with respect to that file.

2. Discretionary Access Control (Cont...)

- An **access control triple** consists of the user, program, and file, with the corresponding access privileges noted for each user.
- This type of access control is used in local, dynamic situations in which the subjects must have the discretion to specify what resources certain users are permitted to access.
- When a user within certain limitations has the right to alter the access control to certain objects, this is termed a **user-directed discretionary access control**.
- An **identity-based access control** is a type of discretionary access control based on an individual's identity.
 - It determines what you can do based on who you are.
 - The user performing an API request is first authenticated and then a check is performed to see if that user is authorized to perform the action they're requesting.
 - In some instances here, a hybrid approach is used, which combines the features of user-based and identity-based discretionary access control.

3. Nondiscretionary Access Control

- Here, a central authority determines which subjects can have access to certain objects based on the **organizational security policy**.
- The access controls are decided based on the **individual's role** in the organization (**Role-based access control**) or the subject's responsibilities and duties (**task-based**).
- It is useful in **organizations frequent personnel changes** occur as the access controls are based on the individual's role or title within the organization. Therefore, these access controls don't need to be changed whenever a new person assumes that role.

3. Nondiscretionary Access Control (Cont...)

- Access control can also be characterized as *context-dependent* or *content-dependent*.
- Context-dependent access control is a **function of factors such as location, time of day, and previous access history**. It is concerned with the environment or context of the data.
- In content-dependent access control, access is determined by the **information contained** in the item being accessed.

How do you deal with the security threats from the Cloud Service Providers themselves (Insider Attackers)?

Can anyone suggest any solution in this context?.

Solution:

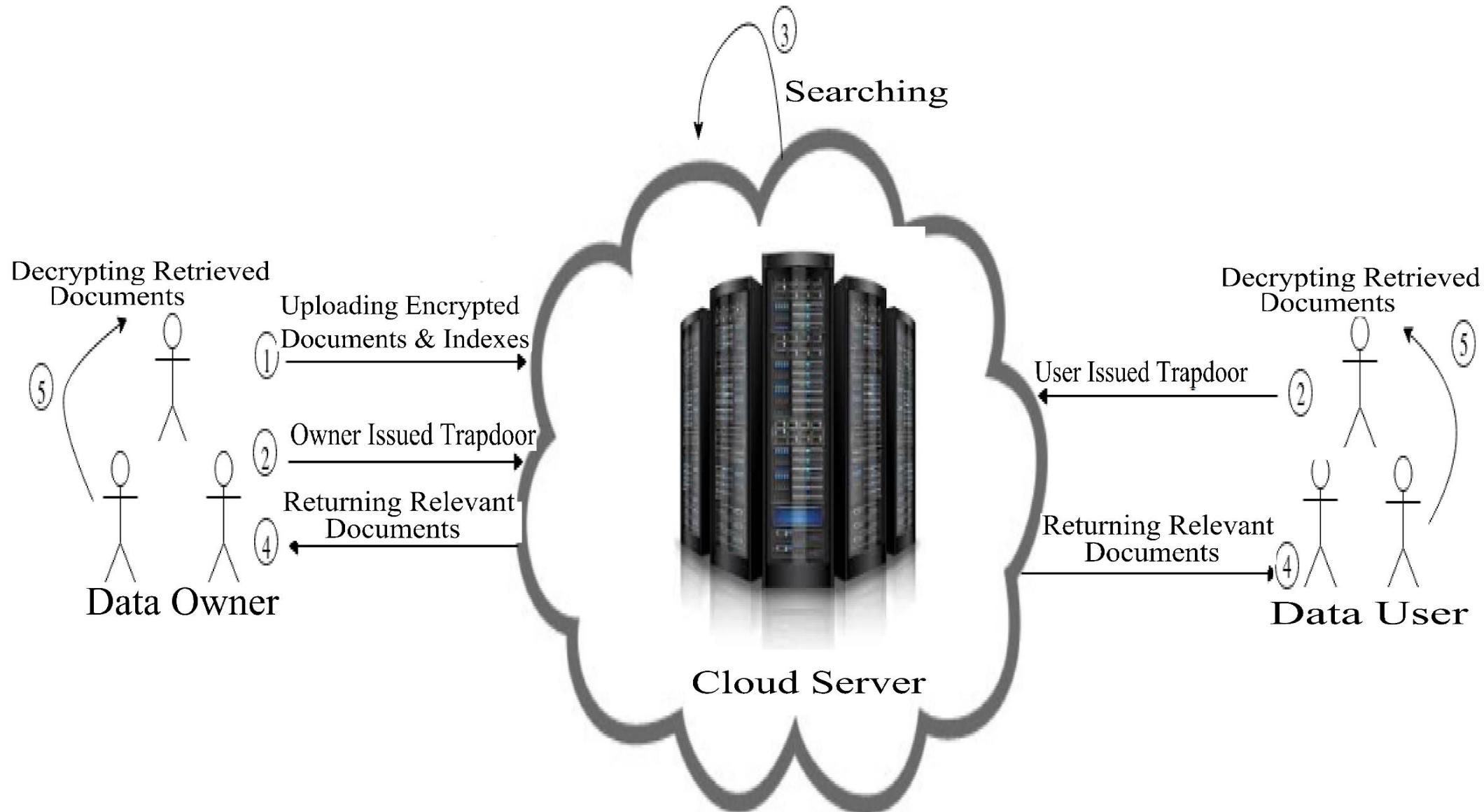
- Store encrypted data at the cloud servers instead of plaintext form. This guarantees security but complicates data retrieval operation.

Does anyone know anything approach that would guarantee data security and yet simplifies the retrieval operation.?.

Solution: Searchable Encryption

It is a cryptographic paradigm that enables the cloud server to perform search operation over encrypted data without knowing anything about plaintext information.

Searchable Encryption Process



Will searchable encryption resolves all security issues because of storing encrypted data at cloud servers.?

No, there are still various security issues associated with the following information disclosure attacks:

- Frequency Analysis Attacks (Frequency leakage).
- Scale Analysis Attacks (Search pattern leakage).
- Rank-order Exploitation Attacks (rank information leakage)
- Access Pattern Exploitation Attacks (Access pattern leakage)
- Keyword Guessing Attacks.

Observe Frequency of Plaintext values of various keywords

Plaintext Index

	D1	D2	D3	D4	D5	Dn
W1	2	1	2	2	0	2
W2	0	1	3	4	5	3
W3	1	0	4	0	1	1
W4	2	1	3	2	1	2
Wm	1	2	1	0	3	2

Observe Frequency of ciphertext values of corresponding keywords

Encrypted Index

	D1	D2	D3	D4	D5	Dn
EW1	1200	1000	1200	1200	101	1200
EW2	101	1000	1500	1900	2100	1500
EW3	1000	101	1900	101	1000	1000
EW4	1200	1000	1500	1200	1000	1200
EWm	1000	1200	1000	101	1500	1200

Assume Order-Preserving Encryption schemes are used to encrypt the plaintext values of keywords in indexes.

Frequency Leakage allows us to infer plaintext keyword from the encrypted values of encrypted index.

-
- The insider attackers mount the various attacks (discussed) and infer plaintext information from the encrypted information with the help of some background knowledge about the stored data.

Mitigating or Preventing the information disclosure attacks:

- We need to combine various privacy-preserving searchable encryption approaches based on **privacy, precision and efficiency** requirements.

Note: Searchable encryption approaches can provide benefits to both the data owners and as well as Cloud service Providers (CSPs).

Availability

- Ensuring high availability is important for both mission-critical and non-mission critical applications, as the downtime implies loss of revenue.
 - Availability can be ensured using two approaches:
 1. High availability for underlying application
 2. Build support for high availability into infrastructure
- 1) High availability for the underlying application involves following three techniques: They are:
- i) **Infrastructure Availability:** It ensures redundancy in infrastructure, such as servers, so that new servers are always ready to replace failed servers.
 - ii) **Middleware Availability:** It deals with middleware redundancy.
 - iii) **Application Availability:** It is achieved using application redundancy.

2) Build Support for the High Availability into the Infrastructure

Two types of support can be built into the infrastructure:

1) Failure detection:

- The cloud infrastructure detects failed application instances, and avoids routing requests to such instances.
- Many CSPs such as **AWS Elastic Beanstalk**, detect when an application instance fails, and avoid sending new requests to the failed instance.
- To detect failures, it is required to monitor for failures: Two techniques are used:

i) Heartbeats:

- The application instance periodically sends a signal (called a heartbeat) to a monitoring service in the cloud.
- If the monitoring service does not receive a specified number of consecutive heartbeats, it may declare that the application instance as failed.

ii) Probes:

- The monitoring service periodically sends a probe, which is a lightweight service request to the application instance.
- If the instance does not respond to a specified number of probes, it may be considered failed.

- **Redirection:** After identifying failed instances, it is necessary to avoid routing new requests to the failed instances.

- HTTP-Redirection: The web server may return a 3xx return together with a new URL to be visited.
- For example, if a user types <http://www.pustakportal.com/> into their browser, the request first be sent to a load-balancing service at Pustak Portal, which may return a return code of 302 with a URL <http://pns5.pustakportal.com> (pns5.pustakportal.com is the address of a server that is currently known to be up.)

2) Application Recovery:

- It is necessary to recover old requests in addition to directing new requests to a server that is up.
- **Checkpoint/Restart:** The cloud infrastructure periodically saves the state of the application. If the application determined to have failed, the most recent check point can be activated and the application can resume from that state.
 - The infrastructure should check point all resources(e.g., system memory)
 - Checkpointing storage will normally require support from the storage or file system, since any updates that were performed have to be rolled back.
 - **Distributed checkpoint/restart:** All processes of distributed application instances are checkpointed, and all instances are restarted from a common check-point if any instance fails.

Service Level Agreement (SLA)

- A cloud service SLA is a document defining the agreement or interaction between the customer and the cloud server provider.
- An SLA must contain list of services, metrics to evaluate (response time, outages per month, N/W and Storage throughput), mechanism to monitor the service, remedies or credits to be given if terms of SLA are not met, expected changes in SLA over time, roles and responsibilities of customer, etc.
- Cloud providers offer the following types of SLA:
 - **Off-the-Shelf SLAs:** This can be found on their website. They offer credits toward the monthly bill for SLA violations. They are non-negotiable and usually unacceptable to enterprises wanting to host critical services on the cloud.
 - **Negotiable SLAs:** These are more expensive, because they are customized for the client.

Service Level Objective (SLO)

- SLO is part of SLA that defines the characteristic of a service in specific and quantifiable terms.
- Some examples:
 - Application must not have more than 15 pending requests at any instant.
 - Response for a read request should initiate within 3 seconds.
 - Data must be stored within the Bangalore and Mumbai data centers.

SLA Aspects and Requirements

There are various considerations that must be specified within an SLA. Some of the key elements that help make a compact SLA are described below:

- Service Availability
 - Data Locations
 - Availability Zones
 - Downtime Credits
 - Credit Initiation
 - Mean Time to Repair
 - Data Protection
 - Data Encryption
 - Regulator Requirements
 - Certifications
 - Scheduled Maintenance Periods
-

Multitenancy – Introduction

- Multi-tenancy is an architecture in which a single instance of a software application serves multiple customers. Each customer is called a tenant. Tenants may be given the ability to customize some parts of the application, such as color of the user interface (UI) or business rules, but they cannot customize the application's code.
- A software-as-a-service ([SaaS](#)) provider, for example, can run one instance of its application on one instance of a database and provide web access to multiple customers. In such a scenario, each tenant's data is isolated and remains invisible to other tenants.

Multitenancy – Introduction

- Multi-tenancy is an architectural pattern
- A single instance of the software is run on the service provider's infrastructure
- Multiple tenants access the same instance.
- In contrast to the multi-user model, multi-tenancy requires customizing the single instance according to the multi-faceted requirements of many tenants.

Multitenancy – key aspects

A Multi-tenants application lets customers (tenants) share the same hardware resources, by offering them one shared application and database instance ,while allowing them to configure the application to fit there needs as if it runs on dedicated environment.

These definition focus on what we believe to be the key aspects of multi tenancy:

- 1.The ability of the application to share hardware resources.
- 2.The offering of a high degree of configurability of the software.
- 3.The architectural approach in which the tenants make use of a single application and database instance.

Multi-tenants Deployment Modes for Application Server

Fully isolated Application server Each tenant accesses an application server running on a dedicated servers.	<p>The diagram shows two separate vertical stacks. The top stack is labeled 'Application Server' and contains two blue circles labeled 'Tenant A'. The bottom stack is also labeled 'Application server' and contains two blue circles labeled 'Tenant B'. Lines connect each tenant's circles to its respective application server.</p>
Virtualized Application Server Each tenant accesses a dedicated application running on a separate virtual machine.	<p>The diagram shows two separate vertical stacks. The top stack is labeled 'Application server' and contains two blue circles labeled 'Tenant A'. The bottom stack is also labeled 'Application server' and contains two blue circles labeled 'Tenant B'. Each stack has a blue box labeled 'Virtual machine' to its right. Lines connect each tenant's circles to its respective virtual machine, which then connects to the application server.</p>
Shared Virtual Server Each tenant accesses a dedicated application server running on a shared virtual machine.	<p>The diagram shows two separate vertical stacks. The top stack is labeled 'Application server' and contains two blue circles labeled 'Tenant A'. The bottom stack is also labeled 'Application server' and contains two blue circles labeled 'Tenant B'. Both stacks point to a single green circle labeled 'Virtual machine' which contains a blue box labeled 'Application server'. Lines connect each tenant's circles to the shared virtual machine, which then connects to the application server.</p>
Shared Application Server The tenant shared the application server and access application resources through separate session or threads.	<p>The diagram shows two separate vertical stacks. The top stack is labeled 'Application server' and contains two blue circles labeled 'Tenant A'. The bottom stack is also labeled 'Application server' and contains two blue circles labeled 'Tenant B'. Both stacks point to a single green circle labeled 'Application Server' which contains three blue circles labeled 'Session thread'. Lines connect each tenant's circles to the shared application server, which then connects to the session threads.</p>

Multi-tenants Deployment Modes in Data Centers

Fully isolated data center The tenants do not share any data center resources	<p>The diagram shows two separate server racks. The top rack is labeled 'Tenant A' and contains a server icon and two database cylinder icons. The bottom rack is labeled 'Tenant B' and also contains a server icon and two database cylinder icons. Blue lines connect the tenant icons to their respective servers, and red lines connect the servers to their respective databases.</p>
Virtualized servers The tenants share the same host but access different databases running on separate virtual machines	<p>The diagram shows a single host represented by a green rectangle containing a server icon. Two separate database cylinders are shown. Tenant A is connected to the host via a blue line and to one database via a red line. Tenant B is connected to the host via a blue line and to another database via a red line. The host connects to the databases via a green line.</p>
Shared Server The tenants share the same server (Hostname or IP) but access different databases	<p>The diagram shows a single host represented by a green rectangle containing a server icon. Two separate database cylinders are shown. Tenant A is connected to the host via a blue line and to one database via a red line. Tenant B is connected to the host via a blue line and to another database via a red line. The host connects to the databases via a green line.</p>
Shared Database The tenants share the same server and database (shared or different ports) but access different schema(tables)	<p>The diagram shows a single host represented by a green rectangle containing a server icon. A single database cylinder is shown. Tenant A is connected to the host via a blue line and to the database via a red line. Tenant B is connected to the host via a blue line and to the same database via a red line. The host connects to the database via a green line. The database is represented as a grid of blue and grey squares.</p>
Shared Schema The tenants share the same server, database and schema (tables). The irrespective data is segregated by key and rows.	<p>The diagram shows a single host represented by a green rectangle containing a server icon. A single database cylinder is shown. Tenant A is connected to the host via a blue line and to the database via a red line. Tenant B is connected to the host via a blue line and to the same database via a red line. The host connects to the database via a green line. The database is represented as a grid of blue and grey squares. A horizontal red bar is present at the bottom of this row.</p>

References

- Ronald L. Krutz, Russell Dean Vines, “Cloud Security: A Comprehensive Guide to Secure Cloud Computing”, Wiley-India, 2010.
- Dinkar Sitaram and Geetha Manjunath. Moving to the Cloud. Syngress (Elsevier) Pub, 2011
- <https://github.com/CryptDB/cryptdb> (Encrypted database)
- <https://dl.acm.org/doi/10.1145/2636328> (Searchable Encryption)
- <https://www.sciencedirect.com/science/article/pii/S0306437917303824> (Searchable encryption)
- <https://www.sciencedirect.com/topics/computer-science/external-attacker> (External attackers)

Thank you all.

Cloud Computing



Module 8 Distributed File System And Hadoop

BITS Pilani

Agenda

- Files and File System
- Distributed File System
- HDFS (Hadoop Distributed File System)
- MapReduce
- MapReduce Problems

Terminology

Service: It is a software entity running on one or more machines and providing a particular type of function to a priori unknown clients.

Server: A server is the service software running on a single machine.

Client: A client is a process that can invoke a service using a set of operations that form its client interface. Sometimes, a lower level interface is defined for the actual cross-machine interaction. Clients implement interfaces suitable for higher level applications or direct access by humans.

File System of a UniProcessor System

- **File:** A file is a collection of data stored in one unit, identified by a filename.
- A **file system** is a subsystem of the operating system that performs file management activities such as **organization, storing, retrieval, naming, sharing, and protection of files.**
- A file system provides file services (Creating a file, reading from a file, writing to a file, etc) to clients.
- The purpose of file system is to **provide long-term storage**. It does so by implementing files - named objects that exist from their explicit creation until their explicit destruction and are immune to temporary failures in the system.

File System of a UniProcessor System (Cont...)

- The primary hardware component a file server controls is a set of secondary storage devices (i.e., magnetic disks) on which files are stored and from which they are retrieved according to the client's requests.
- We often say that a server, or a machine, stores a file, meaning the file resides on one of its attached devices.
- We refer to the file system offered by a uniprocessor, time-sharing operating system as a conventional file system.
- A file system frees the programmer from concerns about the details of **space allocation and layout of the secondary storage device**.

File Types

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

Distributed System and Its File System

Def (Distributed System): It is a collection of loosely connected machines either a mainframe or a workstation interconnected by a communication network.

Def(Distributed File System): A DFS is a file system, whose clients, servers, and storage devices are dispersed among the machines of a distributed system.

- service activity has to be carried out across the network,
- instead of a single centralized data repository there are multiple and independent storage devices.

Note: The configuration and implementation of a DFS may vary.

Note2: There are configurations where servers run on dedicated machines, as well as configurations where a machine can be both a server and a client.

Features of Distributed File System

1. Transparency:

- **Network transparency:** Clients should be able to access remote files using the same set of file operations applicable to local files.

Note: That is, the client interface of a DFS should not distinguish between local and remote files. It is up to the DFS to locate the files and arrange for the transport of the data.

- **Mobility:** It implies that users can log in to any machine in the system; that is, they are not forced to use a specific machine. A transparent DFS facilitates user mobility by bringing the user's environment (e.g., home directory) to wherever he or she logs in.

- **Naming transparency:**

The name of the file should give no hint as to the location of the file. The name of the file must not be changed when moving from one node to another.

- **Replication transparency:** If a file is replicated on multiple nodes, both the existence of multiple copies and their locations should be hidden from the clients.

Features of Distributed File System (Cont...)

2. Performance:

- It is the amount of time needed to satisfy service requests.
- In conventional systems, this time consists of **disk access time and a small amount of CPU processing time**. In a DFS, a remote access has the additional overhead attributed to the distributed structure. This overhead includes the **time needed to deliver the request to a server, as well as the time needed to get the response across the network back to the client**.

Features of Distributed File System (Cont...)

3. **Fault tolerance:** Communication faults, machine failures (of type fail stop), storage device crashes, and decays of storage media are all considered to be faults that should be tolerated to some extent.

- A fault-tolerant system should continue functioning, perhaps in a degraded form, in the face of these failures.
- The degradation can be in performance, functionality, or both but should be proportional, in some sense, to the failures causing it.

4. **Scalability:** The capability of a system to adapt to increased service load is called **scalability**. Systems have **bounded resources** and can become completely saturated under increased load. Regarding a file system, saturation occurs, for example, when a server's CPU runs at very high utilization rate or when disks are almost full.

In DFS, a scalable system should react more gracefully to increased load.

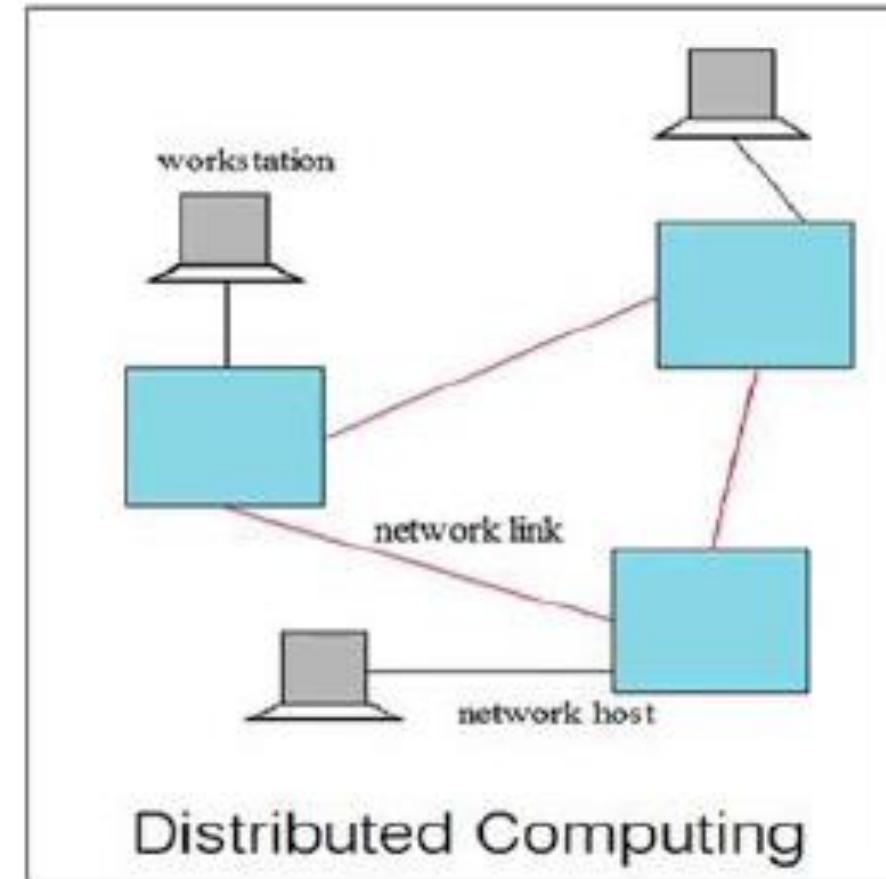
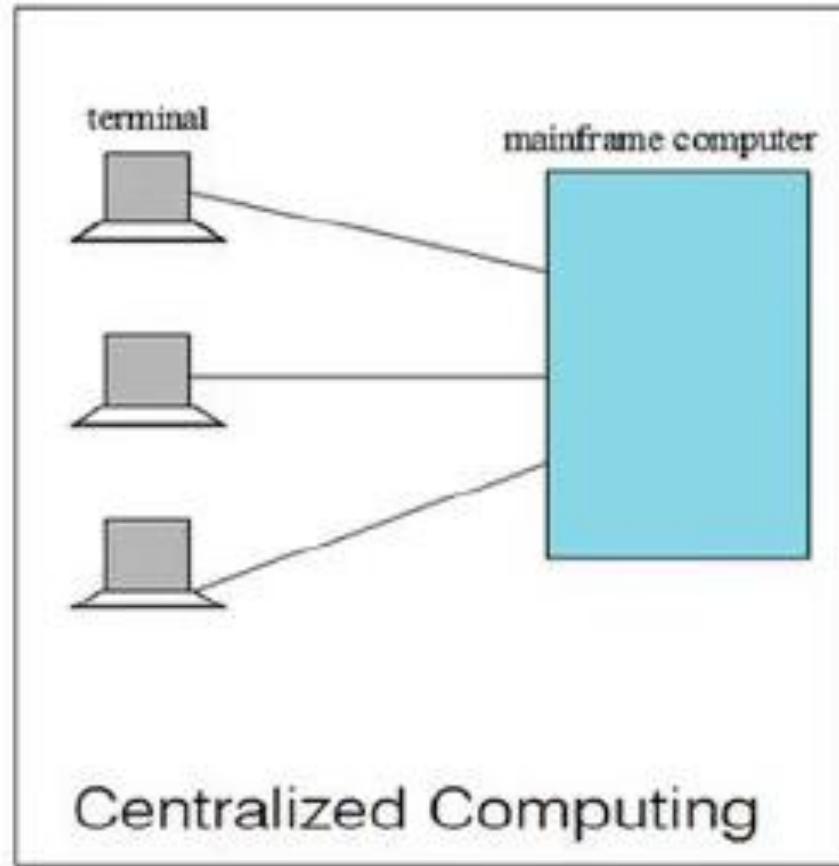
In short, a scalable design should withstand **high-service load, accommodate growth of the user community, and enable simple integration of added resources**.

Note: Fault tolerance and scalability are mutually related to each other.

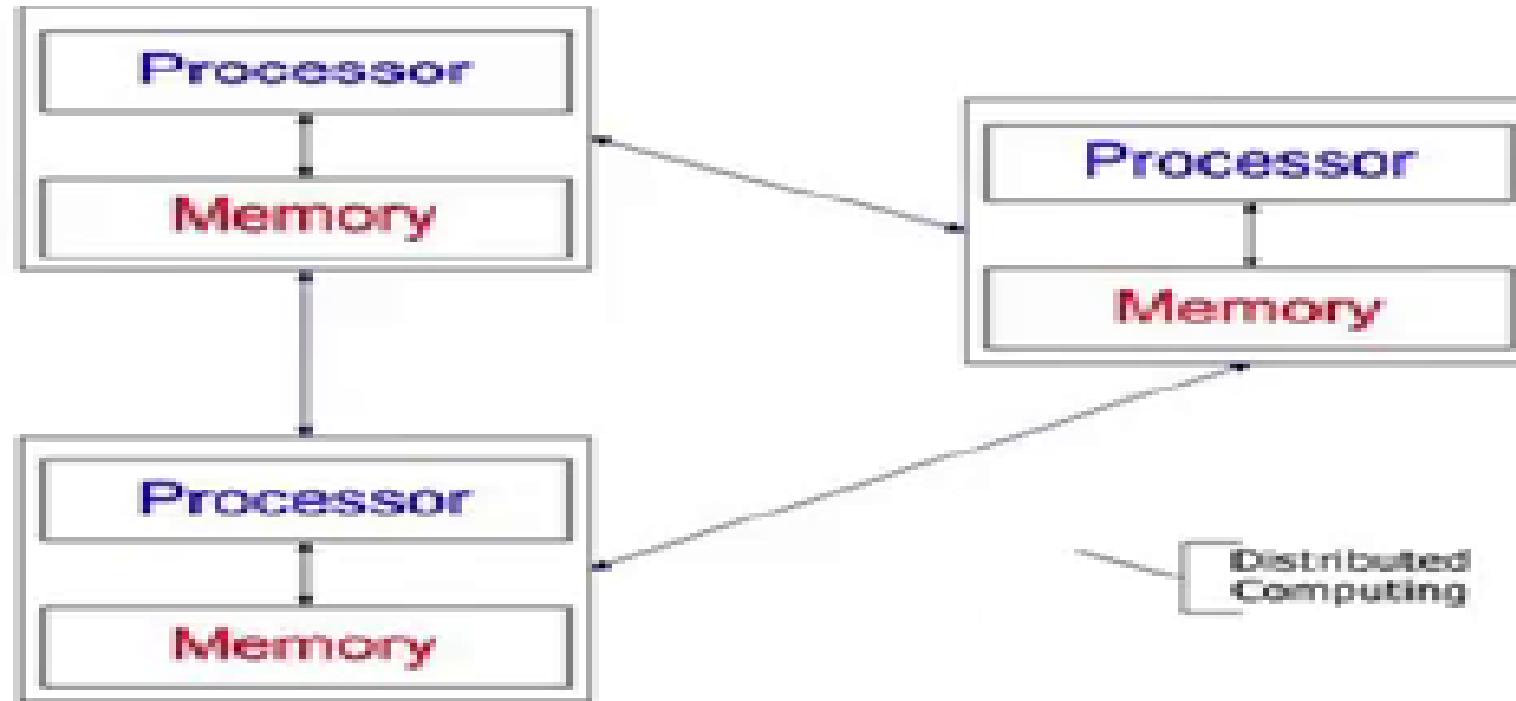
Features of Distributed File System (Cont...)

5. **Simplicity and ease of use:** User interface to the file system be simple and number of commands should be as small as possible.
6. **High reliability:** Probability of loss of stored data should be minimized. System should automatically generate backup copies of critical files.
7. **Data integrity:** Concurrent access requests from multiple users who are competing to access the file must be properly synchronized by the use of some form of concurrency control mechanism. Atomic transactions can also be provided.
8. **Security:** Users should be confident of the privacy of their data.
9. **Heterogeneity:** There should be easy access to shared data on diverse platforms.

Centralized Computing Versus Distributed Computing



Distributed System or Computing



Def: A distributed system is a collection of independent components located on different machines that share messages with each other in order to achieve common goals.

Examples of Distributed Systems are ATMs(Bank Machines), Internet, etc.

Why Distributed Computing?

- Nature of Application
- Performance
- Computing Intensive: The task could consume a lot of time in consuming
- Data Intensive: The task that deals with large size of datasets.
- Robustness:
 - No Single Point of Failure
 - Other nodes can execute the same task executed on the failed node.

Advantages and Disadvantages of Distributed Computing

Major benefits include:

- **Unlimited Horizontal Scaling** - machines can be added whenever required.
- **Low Latency** - having machines that are geographically located closer to users, it will reduce the time it takes to serve users.
- **Fault Tolerance** - if one server or data centre goes down, others could still serve the users of the service.

Disadvantages:

- Data Integration & Consistency
- Network and Communication Failure
- Management Overhead

Computers in a Distributed Computing

- **Workstations:** Computers used by end users to perform computing.
- **Servers:** Computers which provide resources and services
- **Personal Assistance Devices:** Handheld devices connected to the system via a wireless network.

Implementation of DFS

- A DFS can be implemented **as part of a distributed operating system** or, alternatively, **by a software layer** whose task is to manage the communication between conventional operating systems and file systems.

Note: The distinctive features of a DFS are the multiplicity and autonomy of clients and servers in the system.

How do we implement DFS?

Any DFS must look into the following:

1. User Interface
2. Naming
3. Inter process Communication
4. File access mechanism
5. File consistency
6. Synchronization
7. Locking
8. Atomic transactions
9. Error recovery

Examples of DFS are Google File System, HDFS(Hadoop Distributed File System)

Reference:<https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1322&context=theses>

Big Data

What is Big Data?

- **Big Data** is a collection of data that is huge in volume, yet growing exponentially with time.
- It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently.

Few examples of Big Data?.

Stock Market Exchange Data, Social Media, etc.

Challenges of Big Data

- Storage
- Computational Efficiency
- Data Loss
- Cost

Characteristics Of Big Data

- 1. Volume** – It refers to the size of the data that is enormous. Size of data plays a very crucial role in determining value out of data.
- 2. Variety**-: It refers to heterogeneous sources and the nature of data, both structured and unstructured. Data in the form of spreadsheets, tables (RDBMS), emails, photos, videos, monitoring devices, PDFs, audio, etc.
- 3. Velocity** : It refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data. The data flows in from sources like **business processes, application logs, networks, and social media sites, sensors, Mobile devices**, etc. The flow of data is **massive and continuous**.
- 4. Variability** – This refers to the **inconsistency which can be shown by the data at times**, thus hampering the process of being able to handle and manage the data effectively.

Types Of Big Data

-
- 1. Structured Data:** Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. For example, data stored in tables.
 - 2. Unstructured Data:** Any data with unknown form or the structure is classified as unstructured data. For example, heterogeneous data source containing a combination of simple text files, images, videos etc.
 - 3. Semi-structured Data:** It can contain both the forms (structured and unstructured) of data. For example, the data represented in XML files.

Benefits of Processing Big Data:

- Businesses can utilize outside intelligence while taking decisions.
- Early identification of risk to the product/services, if any.
- Better operational efficiency.

Hadoop

- Hadoop is an open source software framework used to develop data processing applications which are executed in a distributed computing environment.
- It offers the cost effective solution to the Big Data Problem.
- It is designed and deployed by Apache Foundation.
- It is one of the best cloud Platforms for Big Data.
- Applications built using HADOOP are run on large data sets distributed across clusters of commodity computers

Hadoop (Cont...)

- Hadoop includes a **Distributed File System (HDFS)** and a **MapReduce framework** to transform and analyze huge data sets.
- An important **characteristic of Hadoop** is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data.
- A Hadoop cluster **scales computation capacity**, storage capacity and IO bandwidth by simply adding commodity servers.

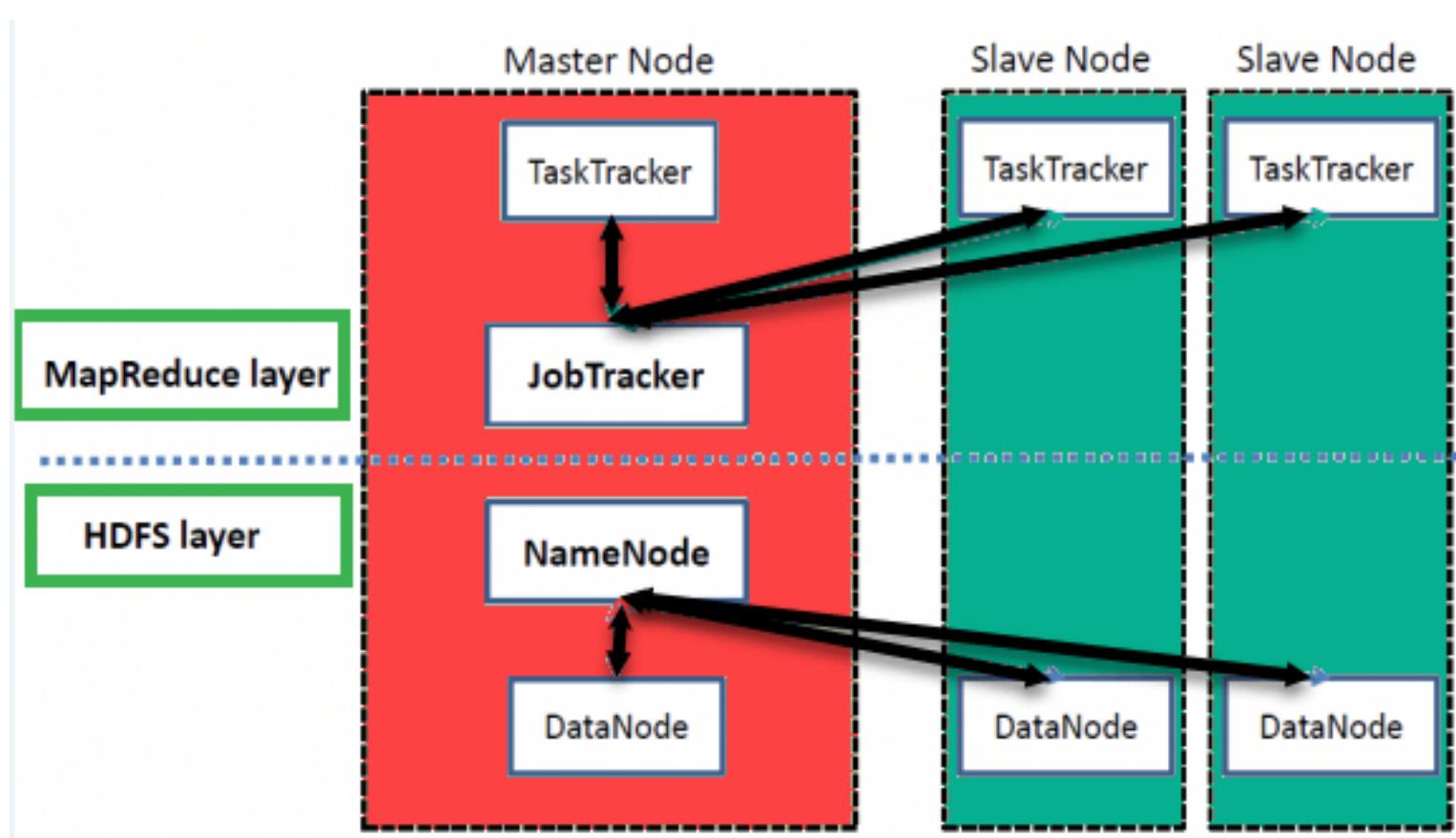
Hadoop Features

- Suitable for Big Data Analysis
- Scalability
- Fault Tolerance
- High Availability
- Cost-Effective
- High Flexibility
- Faster Data Processing

Hadoop Terminology

1. **Job:** a full program – an execution of a Mapper and Reducer across data set
2. **Task:** An execution of a mapper or reducer on a slice of data.
3. **Task Attempt:** A particular instance of an attempt to execute a task on a machine

Hadoop Architecture



Hadoop Master-Slave Architecture for data storage and distributed data processing using **HDFS** and **Map Reduce**

1. HDFS (Hadoop Distributed File System)

- Hadoop File System was developed using distributed file system design.
- It allows us to store large data across multiple nodes in a Hadoop cluster. Each file on HDFS is stored as sequence of blocks all of which are of the same size except the last block.
- HDFS stores file system **metadata and application data** separately.
- **Metadata on a dedicated server**, called the NameNode. **Application data** are stored on other servers called **DataNodes**.
- All servers are fully connected and communicate with each other using TCP-based protocols.
- In HDFS, the file content is replicated on multiple DataNodes for reliability. This strategy has the added advantage that data transfer band-width is multiplied, and there are more opportunities for locating computation near the needed data.
- By default a **file's replication factor is three**

Ref: <https://storageconference.us/2010/Papers/MSST/Shvachko.pdf> (HDFS)

2. MapReduce

- It is a **distributed programming model** for processing large datasets of key-value pairs in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner..
- It is conceived at Google Inc.
- MapReduce consists of two distinct tasks – **Map** and **Reduce**.
- It works by breaking the processing into two phases.: the **map phase and the reduce phase**.
- Each phase has key-values pairs as input and output, the types of which may be chosen by the programmer.
- As the name MapReduce suggests, the reducer phase takes place after the mapper phase has been completed.
- It can be implemented in any programming languages, e.g., Java, Python, etc.
- It manages communications, data transfers, parallel execution across distributed servers

Hadoop Architecture (Cont...)

MasterNode/NameNode:

- Stores metadata for the files, like the directory structure of a typical FileSystem.
- It keeps track of where across the cluster the file data is kept.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc.
- Handles creation of more replica blocks when necessary after a DataNode failure

DataNode/ Slave Node:

- Stores the actual data in HDFS and they connect to the NameNode on startup.
- They respond to requests from the NameNode for filesystem operations. Client applications can directly talk to the DataNodes
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has.
- NameNode replicates blocks 2x in local rack, 1x elsewhere

Hadoop Architecture (Cont...)

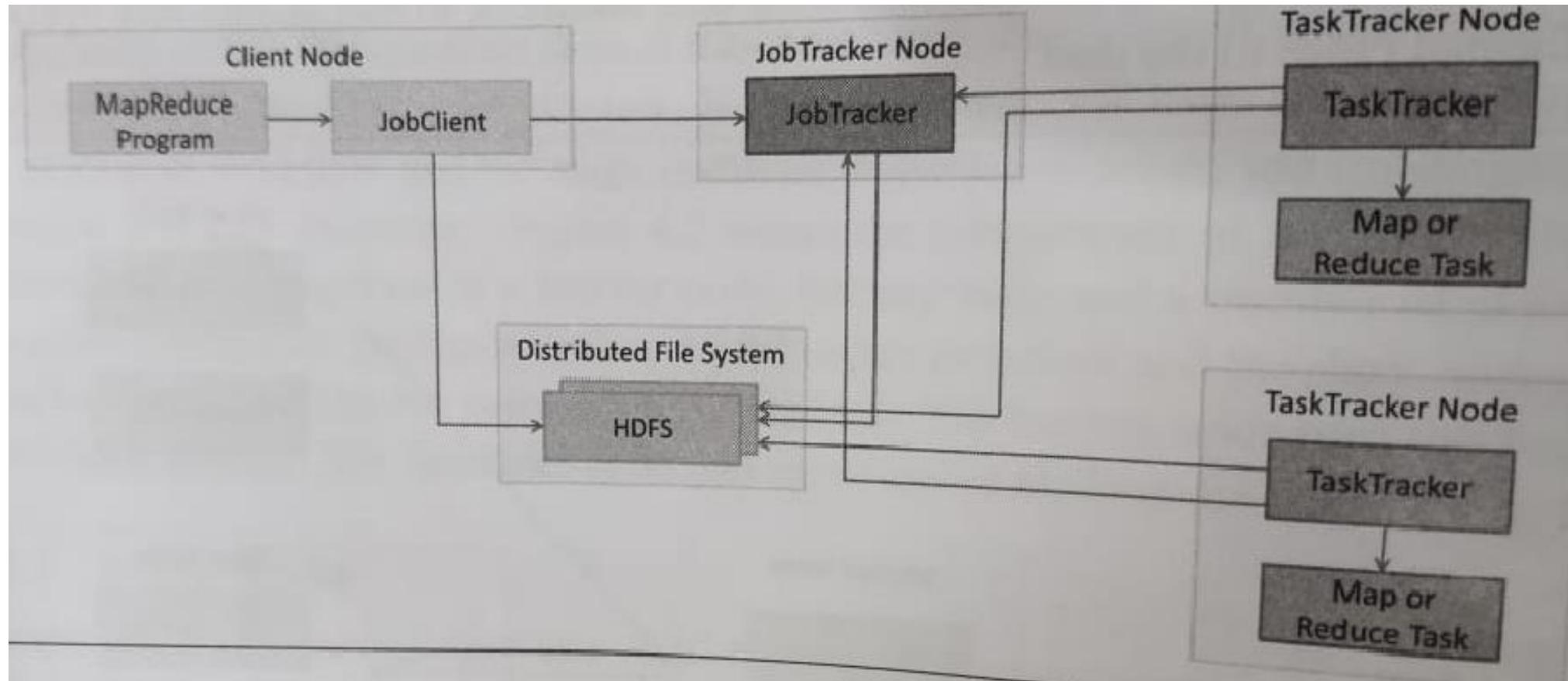
The Job Tracker:

- It is the service (runs on Master node) within Hadoop that distributes MapReduce tasks to specific nodes in the cluster(nodes that have data).
- It is responsible for scheduling and monitoring MapReduce jobs
- Responds to client request for job submission and status

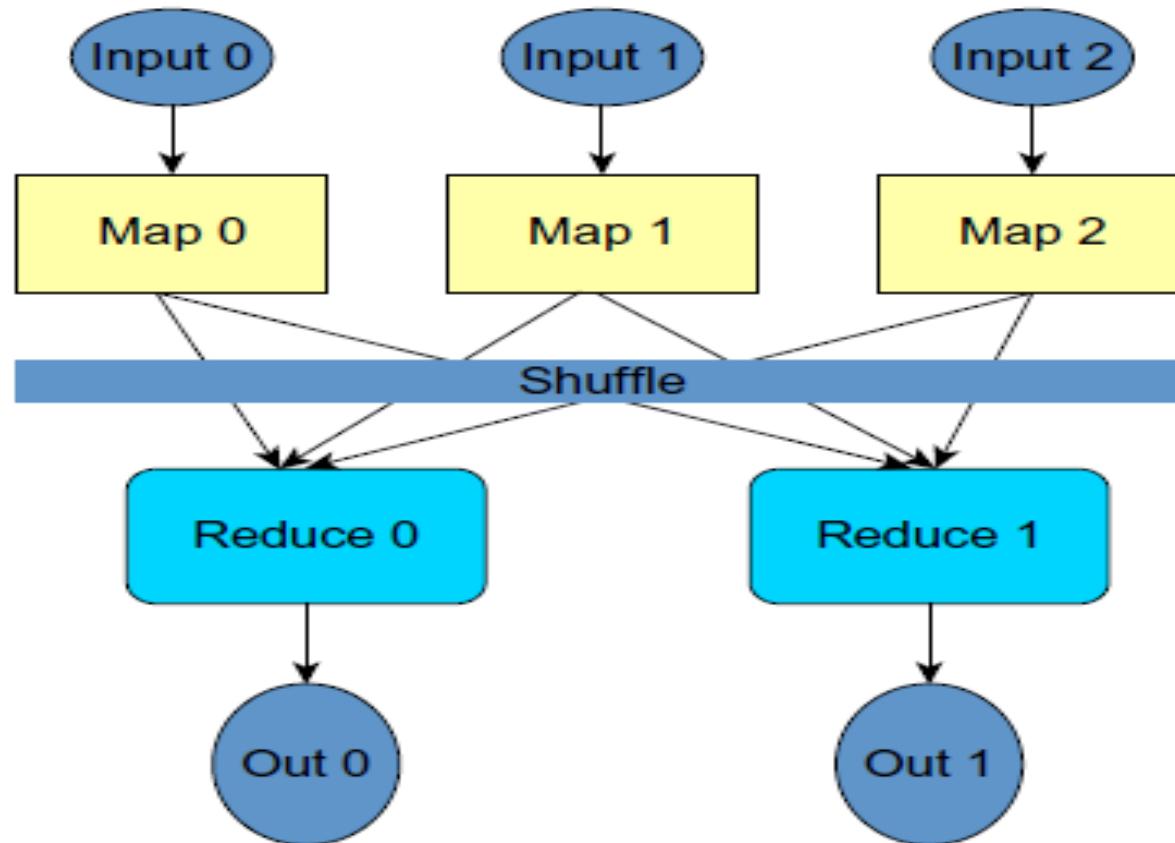
The Task Tracker:

- The TaskTrackers are the workers (nodes).
- Workers that accepts Map, Reduce and Shuffles tasks from job tracker, launches them and keeps track of their progress, reports the same to job tracker.
- Keeps track of resource usage of tasks and kills the tasks that overshoots their memory limits

Hadoop MapReduce Job Execution



Data Flow in MapReduce



Inputs and outputs of Both Mappers and Reducers

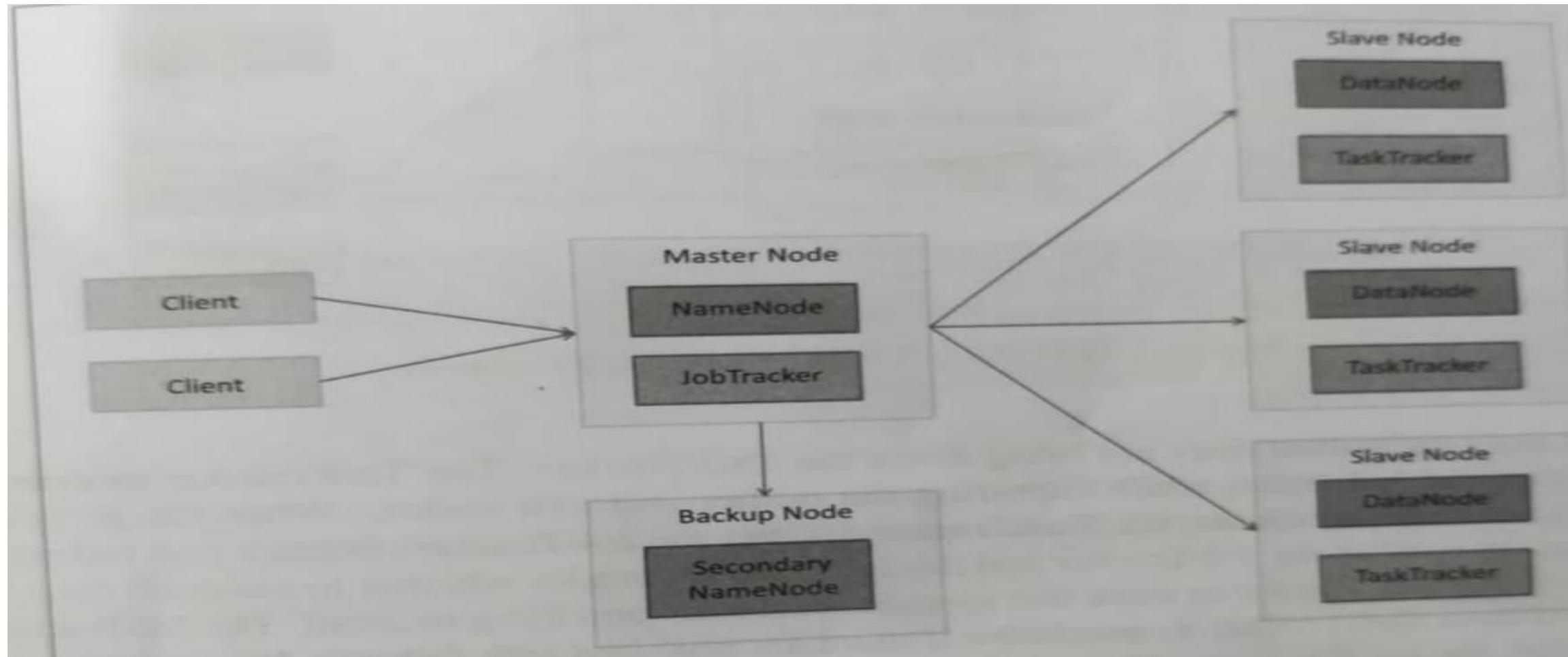
- i) **Mapper**, where a block of data is read and processed to produce **key-value pairs** as intermediate outputs.
Note: The input of the mapper is also key-value pair.
- ii) The **Reducer** receives the key-value pair from multiple map jobs. Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

Advantages of MapReduce

Advantages of MapReduce

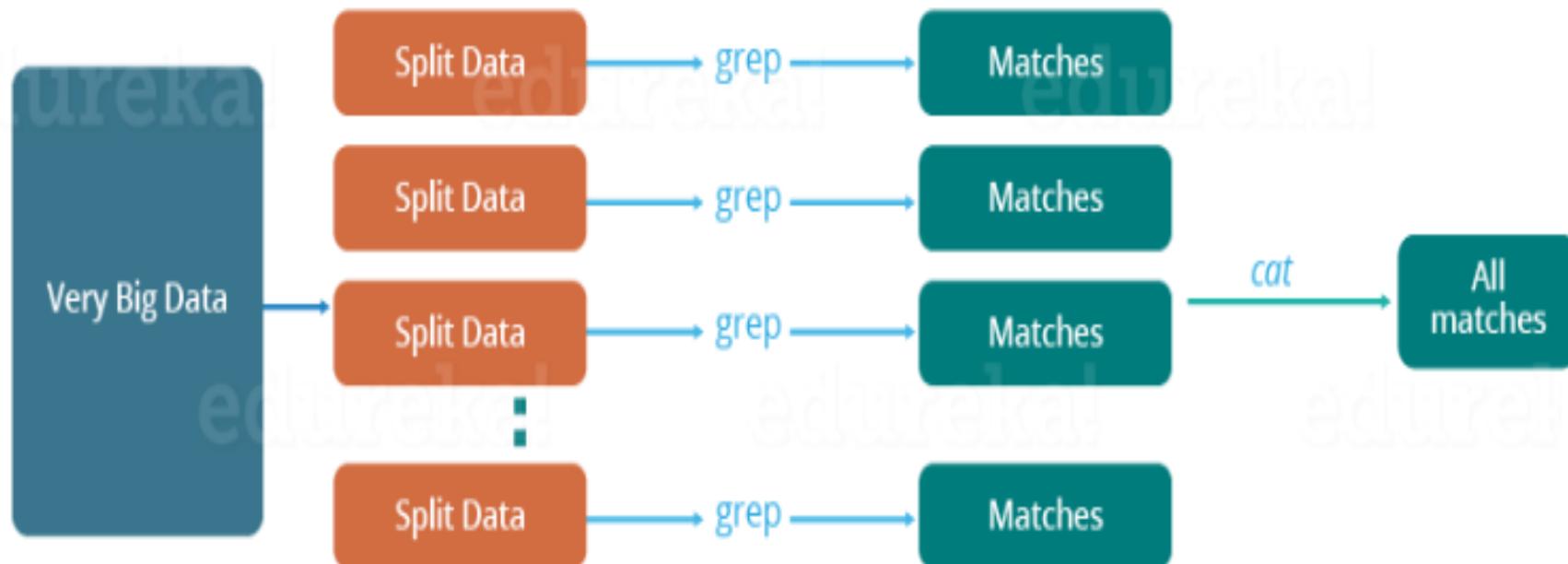
1. Parallel Processing
2. Data Locality

Hadoop Cluster



Traditional Way of Solving the Big problem

Task: Calculate the day having the highest temperature in each year from the log containing the daily average temperature of the years from 2000 to 2015..



Ref: <https://www.edureka.co/blog/mapreduce-tutorial/>

Challenges to deal with the traditional approach

Problems to dealt with:

- **Critical path problem** (if, any of the machines delay the job, the whole work gets delayed.)
- **Reliability problem** (What if, any of the machines which are working with a part of data fails?)
- **Equal split issue** (how to equally divide the data such that no individual machine is overloaded or underutilized.)
- **The single split may fail:** If any of the machines fail to provide the output, I will not be able to calculate the result. So, there should be a mechanism to ensure this fault tolerance capability of the system.
- **Aggregation of the result:** There should be a mechanism to aggregate the result generated by each of the machines to produce the final output

To overcome the above challenges:

MapReduce framework allows us to perform such parallel computations without bothering about the issues like **reliability, fault tolerance, load balancing, etc.** Therefore, MapReduce gives us the flexibility to write code logic without caring about the design issues of the system

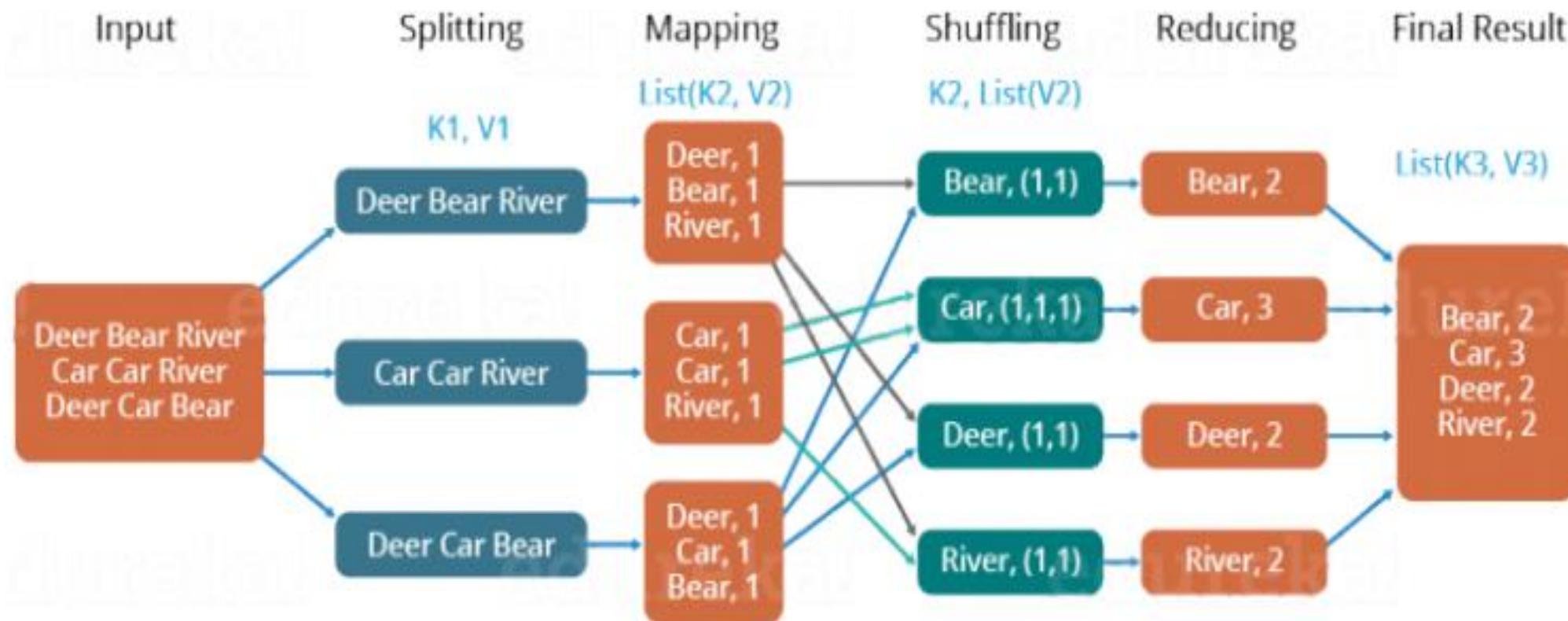
A Word Count Example of MapReduce

Task is to find out the count of each unique keyword of the given file.

File content:

Dear, Bear, River, Car, Car, River, Deer, Car, Bear

MapReduce Word Count Process



Another Example of MapReduce: Finding Maximum Closing Price of each stock symbol

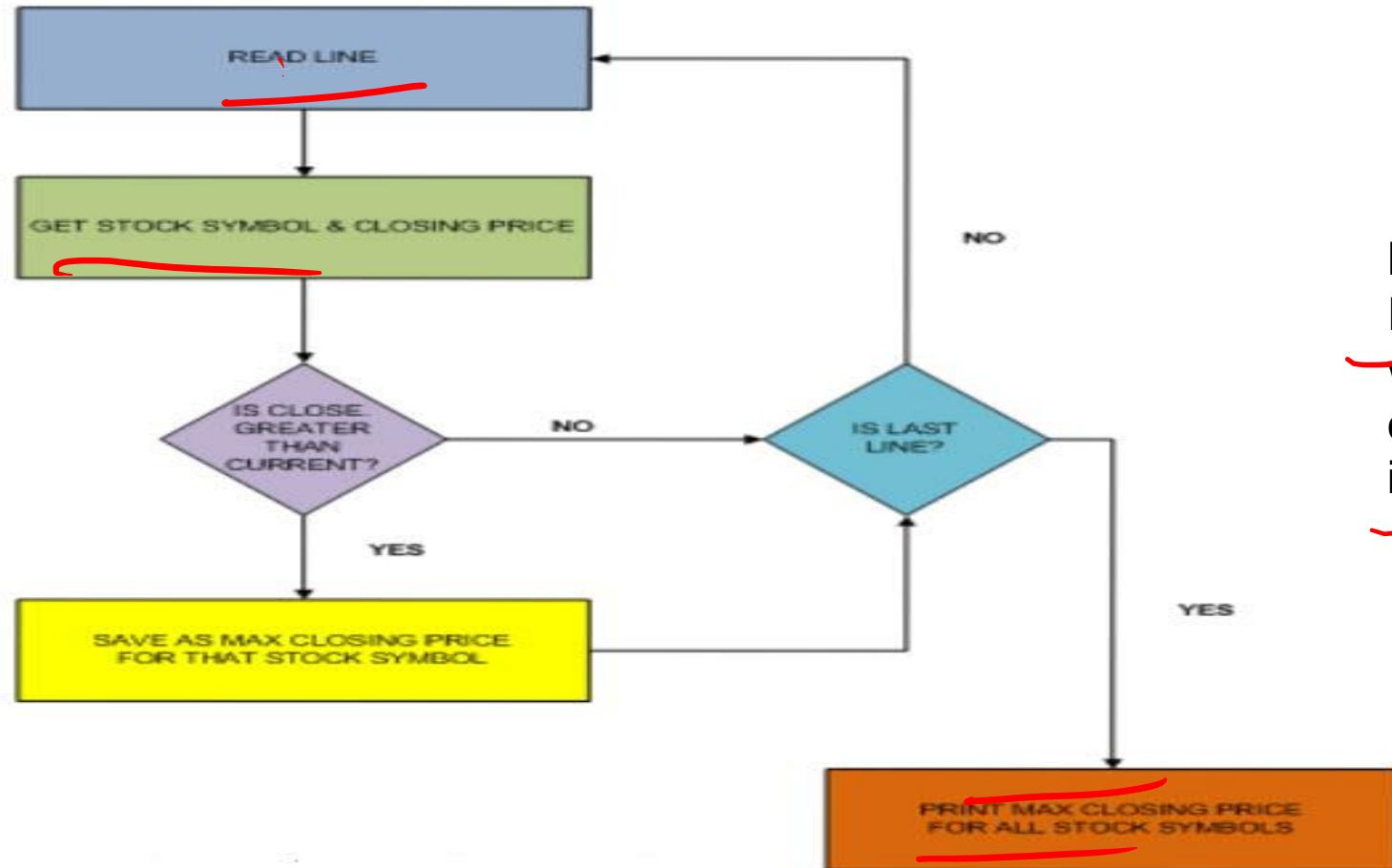
Task: For every stock symbol in the dataset, we would like to find out its maximum closing price across several days.

Input Dataset:

```
1 ABCSE,B7J,2010-01-20,8.93,9.09,8.93,9.09,119300,9.09
2 ABCSE,B7J,2010-01-19,8.99,9.04,8.91,9.02,132300,9.02
3 ABCSE,B7J,2010-01-15,8.98,9.00,8.90,8.99,78000,8.99
4 ABCSE,B7J,2010-01-14,9.03,9.03,8.93,9.01,142700,9.01
5 ABCSE,B7J,2010-01-13,8.96,9.00,8.92,8.98,101600,8.98
6 ABCSE,B7J,2010-01-12,8.91,8.95,8.86,8.91,70600,8.91
7 ABCSE,B7J,2010-01-11,8.99,9.00,8.95,8.97,159500,8.97
8 ABCSE,B7J,2010-01-08,8.90,8.95,8.88,8.94,153100,8.94
9 ABCSE,B7J,2010-01-07,8.91,8.94,8.88,8.94,120700,8.94
```

Description of each line of input: For the symbol B7J. for date 2010-01-20 and we have the opening price, high, low close price, volume etc.

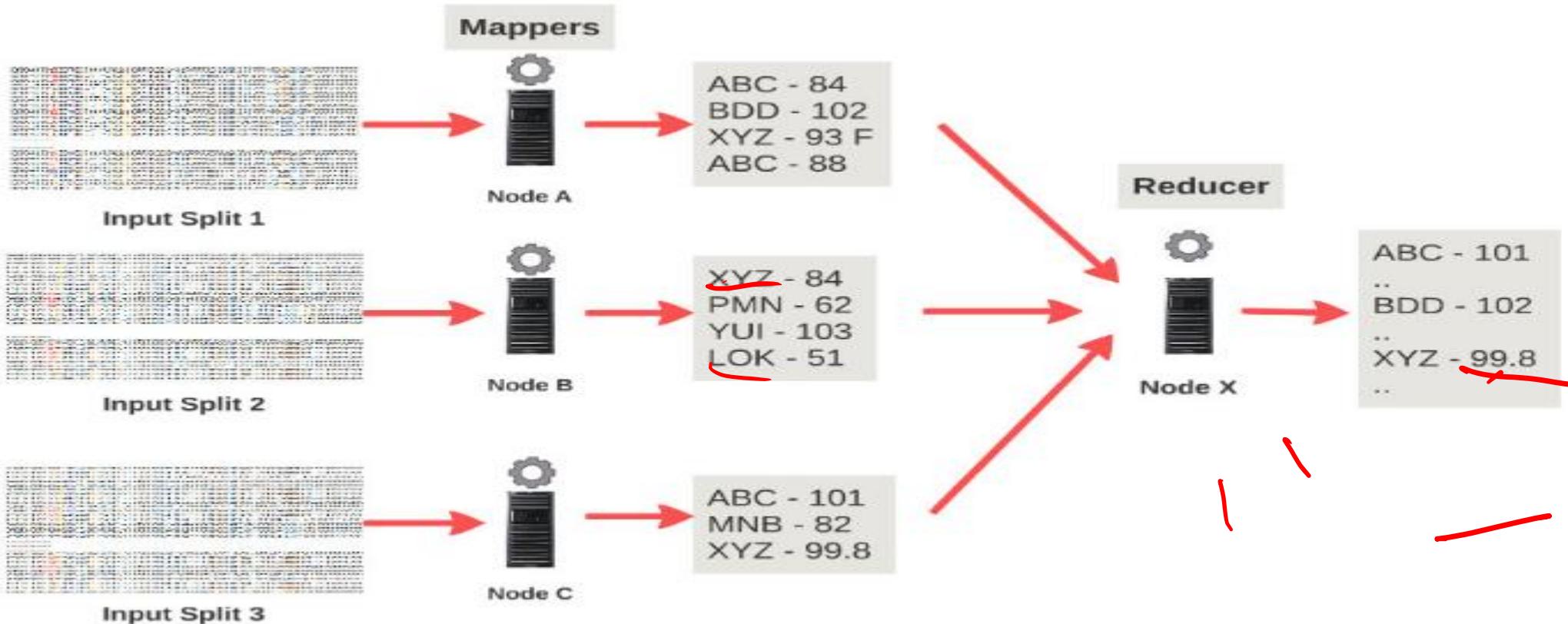
Sample Algorithm (Traditional way) to Solve the Problem



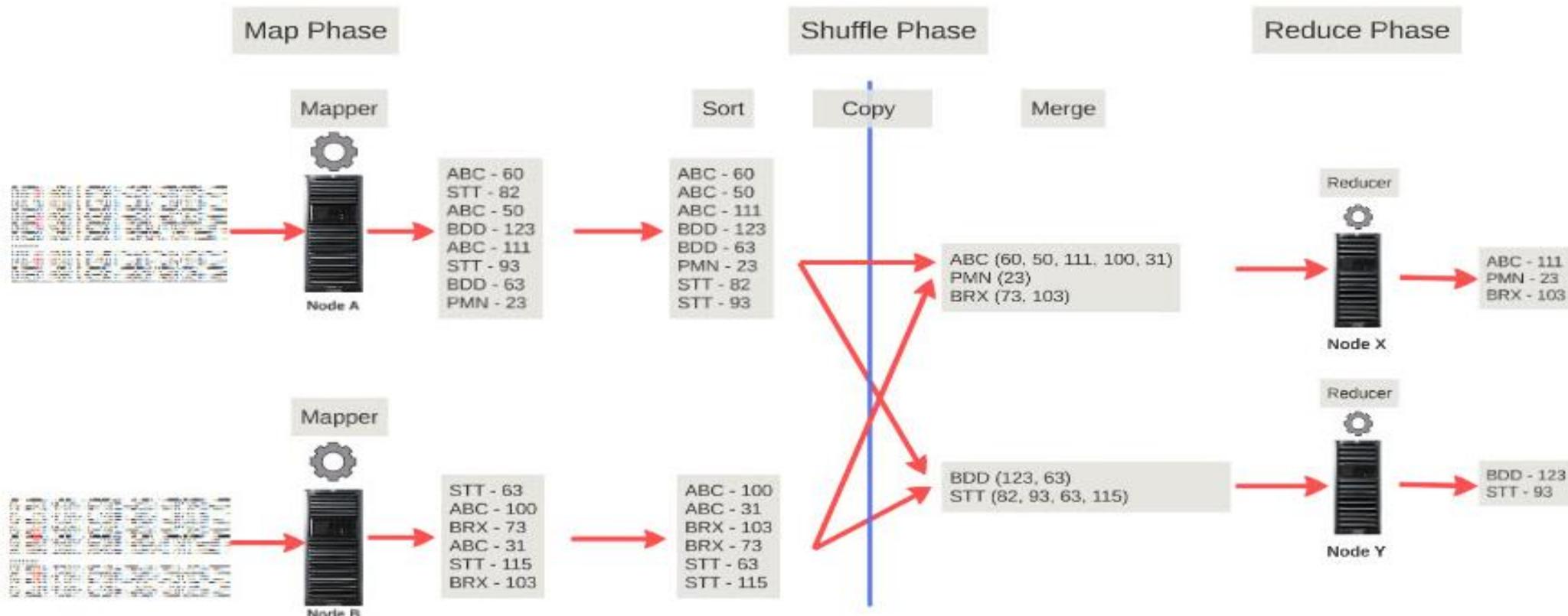
Problem:

If you have a huge dataset you will have extremely long computation time which is not ideal.

MapReduce Solving: Mapper Phase



MapReduce Solving with Multiple Reducers



Another Example: Find the total sales of each book

- The input is a log file that lists the sales of every book from different dealers (one book per line)

(0, "ISBN1234, name of book1, author, dealer name1, location, 10, ...")

(101, "ISBN3245, name of book2, author, dealer name, location, 20,")

(250, "ISBN1234, name of book1, author, dealer name2, location, 110, ...")

...

(1189, "ISBN9999, name of book1111, author, dealer name, location, 32")

Map Phase and Reduce Phase

(ISBN1234, 10)

(ISBN3245, 20)

(ISBN1234, 110)

...

(ISBN9999, 32)

...

MapPhase: Output

(ISBN1234, [10, 110])

(ISBN3245, [20])

...

(ISBN9999, [32, 22, 112])

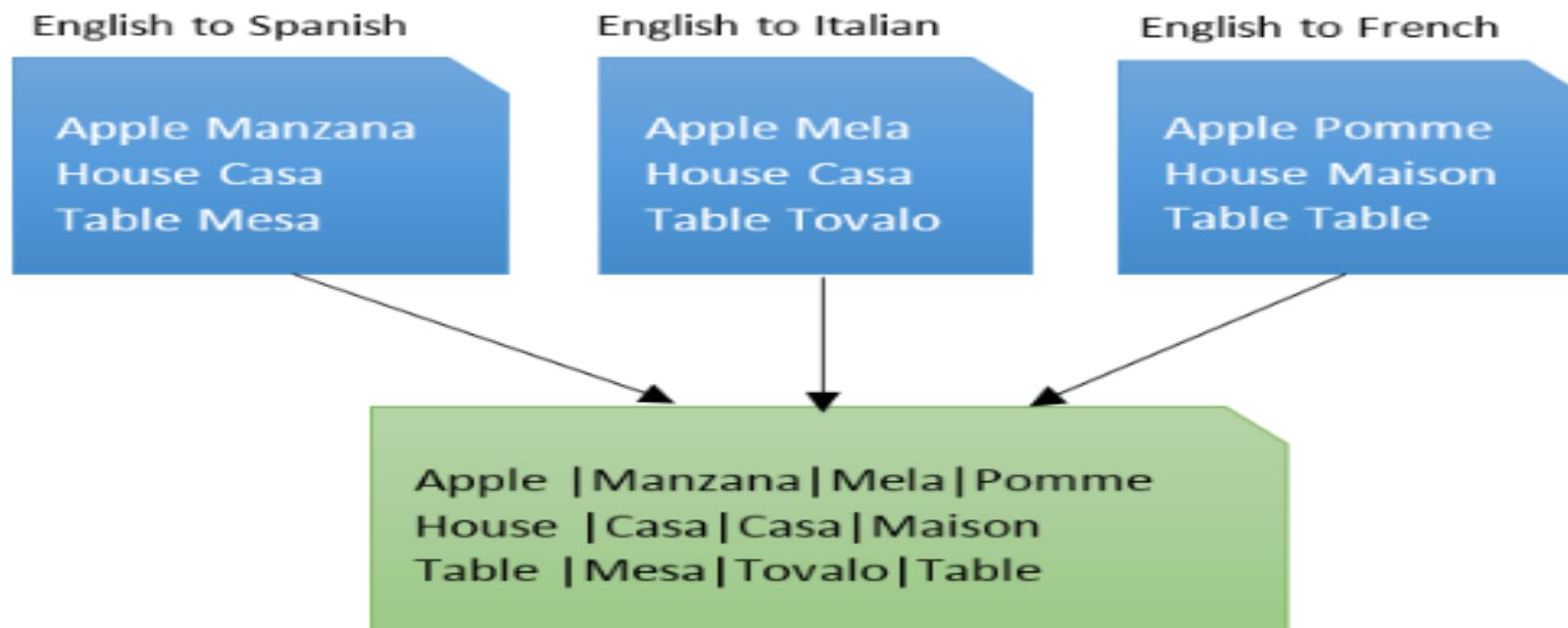
Reducer Phase: Input

?

Reducer Phase: Output

Another Example of MapReduce: Combining Dictionaries

- Take a set of translation dictionaries, **English-Spanish**, **English-Italian**, **English-French**, and create a dictionary file that has the **English word** followed by all the **different translations separated by the pipe (|)** character.



References

1. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “Google File Systems”, SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, Pages 29–432003, October 2003(<https://dl.acm.org/doi/10.1145/1165389.945450>)
2. Tom White, “Hadoop: The definitive Guide”, Third Edition, 2012.
3. Ananth K. Rao, “The DFS distributed file system: Design and implementation”, Thesis, RIT Scholar Works, 1989.