Cloud Computing
CS G527

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

BITS Pilani

What is Datacenter?

Who need Datacenters?

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 teleco mmunications and storage systems..

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Who need Datacenters?

Ans:

- **Any entity that generates or uses data has the need for data centers** including govern ment agencies, educational bodies, telecommunication companies, financial institutions, re tailers 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 service s or loss of customer satisfaction and revenue.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 st ill serve the users of the service.

Disadvantages:

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

Branch Office
Desktop PC
Laptop PC
Control Node
Exchange 2003 Server
Certificate Authority
Web Server
Active Directory Global Catalog Server
SharePoint Server
CRM Server
Perimeter Network

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 Scienece/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.

Parallel Applications
Sequential Applications
Parallel Programming Environments
Cluster Middleware
(Single System Image and Availability Infrstructure)
WorkStation 1
WorkStation 2
WorkStation 3
WorkStation 4
WorkStation 5
Communication switch

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

- 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

Cloud Computing

SEWP ZG527

Motivation

1. Web Scale Problems
2. Web 2.0 and Social Networking
3. Information Explosion
4. Mobile Web

Technology Advances

What is 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).

What is 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

Drivers for the new Platform

http://blogs.technet.com/b/yungchou/archive/2011/03/03/chou-s-theories-of-cloud-computing-the-5-3-2-principle.aspx

Cloud Summary

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

BITS Pilani

Cloud Computing

SEWP ZG527

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.

3-4-5 rule of Cloud Computing

NIST specifies 3-4-5 rule of Cloud Computing

3 cloud service models or service types for any cloud platform

4 deployment models

5 essential characteristics of cloud computing infrastructure

Characteristics of Cloud Computing

- 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.

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.

4 Deployment Models

3. Hybrid Cloud

The cloud infrastructure is a composition of two or more clouds (private or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability

4 Deployment Models

4. Community Cloud

Community Clouds are when an 'infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise' according to NIST.
A community cloud is a cloud service shared between multiple organizations with a common tie/goal/objective.
E.g. OpenCirrus

BITS Pilani

Cloud Computing

SEWP ZG527

3 Cloud Service Models

Software as a Service (SaaS)

Software as a service features a complete application
offered as a service on demand.
A single instance of the software runs on the cloud and services multiple end users or client organizations.
E.g. salesforce.com , Google Apps

Platform as a Service

Platform as a service encapsulates a layer of software and provides it as a service that can be used to build higher-level services.

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.

Infrastructure as a Service

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

Servers, storage systems, switches, routers , and other systems are pooled and made available to handle workloads that range from application components to high-performance computing applications.

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

Cloud Providers 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**

Management of Virtualized Resources

Distributed Management of Virtual Machines

Reservation-Based Provisioning of Virtualized Resources

Provisioning to Meet SLA Commitments

The Cloud OS

The cloud OS, the main component of an IaaS cloud architecture, is organized in three layers: drivers, core components, and high-level tools.

The cloud operating system is responsible for:

1. managing the physical and virtual infrastructure,
2. orchestrating and commanding service provisioning and deployment
3. providing federation capabilities for accessing and deploying virtual resources in remote cloud infrastructures

BITS Pilani

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

13 © Novell Inc. All rights reserved

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 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 infra-structure.
- 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

Hosted Architecture

Bare-Metal (Hypervisor) Architecture

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 content 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

High-level architecture of VMware ESX 3i server

VM virtualization:

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

Xenserver Architecture

- Guest VMs are called Domains.
- Domain 0 is a specialy privileged VM that access to the physical hardware.
- It makes use of paravirtulization for I/O virtulization; (I/O requsts 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-hyphervisor
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?

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3, UGC Act

Current Problem the Industry is facing

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

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 any where.
- 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.

BITS Pilani, Deemed to be University under Section 3, UGC Act

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.

BITS Pilani, Deemed to be University under Section 3, UGC Act

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

```
FROM ubuntu:12.04

RUN apt-get update

# Make it easy to install PPA sources
RUN apt-get install -y python-software-properties

# Install Oracle's Java (Recommended for Hadoop)
# Auto-accept the license
RUN add-apt-repository -y ppa:webupd8team/java
RUN apt-get update
RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
RUN apt-get -y install oracle-java7-installer
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 Containization?

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)

BITS Pilani

Cloud Computing

SEWP ZG527

Technology made cloud possible

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

What does Virtualization do?

- Virtualization allows multiple operating system instances to run concurrently on a single computer
- It is a means of separating hardware from a single operating system.
- Each “guest” OS is managed by a Virtual Machine Monitor (VMM), also known as a hypervisor.
- Because the virtualization system sits between the guest and the hardware, it can control the guests’ use of CPU, memory, and storage, even allowing a guest OS to migrate from one machine to another.
- Instead of purchasing and maintaining an entire computer for one application, each application can be given its own operating system, and all those operating systems can reside on a single piece of hardware.
- Virtualization allows an operator to control a guest operating system’s use of CPU, memory, storage, and other resources, so each guest receives only the resources that it needs.

Changes after Virtualization

Before Virtualization

- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure

After Virtualization

- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines

Virtualization Architecture

- OS assumes complete control of the underlying hardware.
- Virtualization architecture provides this illusion through a hypervisor/VMM.
- Hypervisor/VMM is a software layer which:
  - Allows multiple Guest OS (Virtual Machines) to run simultaneously on a single physical host
  - Provides hardware abstraction to the running Guest OSs and efficiently multiplexes underlying hardware resources

BITS Pilani

Cloud Computing

SEWP ZG527

Hypervisor

A thin layer of software that generally provides virtual partitioning capabilities which runs directly on hardware, but underneath higher-level virtualization services. Sometimes referred to as a “bare metal” approach.

Hypervisor Design Goals

- Isolation
  - Security isolation
  - Fault isolation
  - Resource isolation
- Reliability
  - Minimal code base
  - Strictly layered design
  - Not extensible
- Scalability
  - Scale to large number of cores
  - Large memory systems

How Hypervisor goals are achieved?

- Partitioning Kernel
  - “Partition” is isolation boundary
  - Few virtualization functions; relies on virtualization stack
- Very thin layer of software
  - Microkernel
  - Highly reliable
  - Basis for smaller Trusted Computing Base (TCB)
- No device drivers
  - Drivers run in a partition
- Well-defined interface
  - Allow others to create support for their OSes as guests

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

CPU Sharing

- VMM or Hypervisor provides a virtual view of CPU to VMs.
- In multi processing, CPU is alloted to the different processes in form of time slices by the OS.
- Similarly VMM or Hypervisor allots CPU to different VMs.

Memory Sharing

- In Multiprogramming there is a single level of indirection maintained by Kernel.
- In case of Virtual Machines there is one more level of indirection maintained by VMM

IO Sharing

- Device needs to use Physical Memory location.
- In a virtualized environment, the kernel is running in a hypervisor-provided virtual address space
- Allowing the guest kernel to convey an arbitrary location to device for writing is a serious security hole
- Each device defines its own protocol for talking to drivers

Thanks!!!

Queries?

BITS Pilani

Cloud Computing

SEWP ZG527

Approaches for 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

13 © Novell Inc. All rights reserved

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.
- 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.

Paravirtualization

- “Paravirtualization,” found in the XenSource, open source Xen product, attempts to reconcile these two approaches. Instead of emulating hardware, paravirtualization uses slightly altered versions of the operating system which allows access to the hardware resources directly as managed by the hypervisor.
- This is known as hardware-assisted virtualization, and improves performance significantly.
- 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.
- In this way, paravirtualization can be thought of as a low-overhead full virtualization

SKI Virtualization

- Single Kernel Image (SKI),
- Single Kernel Image (SKI), in which the host OS spawns additional copies of itself. This kind of virtualization can be found in Swsoft Virtuozzo and Sun Solaris, Zones. SKI can be thought of as “lightweight” virtualization.
- While this approach avoids the performance problems with pure emulation, it does so at the expense of flexibility.
- It is not possible, for instance, to run different versions or even different patch levels of a particular operating system on the same machine.
- Whatever versions exist in the host, that same software will be provided in the guest. SKI also sacrifices the security and reliability provided by other virtualization methods.

x86 Hardware Virtualization

- For Industry-standard x86 systems, the two approaches typically used with software-based partitioning are
  - hosted and
  - hypervisor architectures
- A hosted approach provides partitioning services on top of a standard operating system and supports the broadest range of hardware configurations.
- In contrast, a hypervisor architecture is the layer of software installed on a clean x86-based system (hence it is often referred to as a “bare metal” approach). Since it has direct access to the hardware resources, a hypervisor is more efficient than hosted architectures, enabling greater scalability, robustness and performance

x86 Hardware Virtualization

Hosted Architecture

Bare-Metal (Hypervisor) Architecture

Advantages of Virtualization

- Instant provisioning - fast scalability
- Live Migration is possible
- Load balancing and consolidation in a Data Center is possible.
- Low downtime for maintenance
- Virtual hardware supports legacy operating systems efficiently
- Security and fault isolation

Issues to be aware of

- **Software licensing**

  One of the most significant virtualization-related issues to be aware of is software licensing. Virtualization makes it easy to create new servers, but each VM requires its own separate software license. Organizations using expensive licensed applications could end up paying large amounts in license fees if they do not control their server sprawl.

- **IT training**

  IT staff used to dealing with physical systems will need a certain amount of training in virtualization. Such training is essential to enable the staff to debug and troubleshoot issues in the virtual environment, to secure and manage VMs, and to effectively plan for capacity.

- **Hardware investment**

  Server virtualization is most effective when powerful physical machines are used to host several VMs. This means that organizations that have existing not-so-powerful hardware might still need to make upfront investments in acquiring new physical servers to harvest the benefits of virtualization

Issues to be aware of

- Performance can be a concern, especially for in-band deployments, where the virtualization controller or appliance can become a bandwidth bottleneck.
- Interoperability among vendor products is still evolving.
- Failure of the virtualization device, leading to loss of the mapping table.

Thanks!!!

Queries?

BITS Pilani

Cloud Computing

SEWP ZG527

docker

Current Problem the Industry is facing

A shipping container system for applications

Dockers

- 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 cloud portability by running the same applications in different virtual environments.
- Containers are scalable and safer to use and deploy as compared to regular approaches.

Virtual Machines

- Virtual machines are used extensively in cloud computing.
- Isolation and resource control have continually been achieved through the use of virtual machines.
- Virtual machine loads a full OS with its own memory management and enable applications to be more efficient and secure while ensuring their high availability.

Traditional Stack

Virtualized Stack

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?

Docker Container Lifecycle ……

- The Life of a Container
  - Conception
    - **BUILD** an Image from a Dockerfile
  - Birth
    - **RUN** (create+start) a container
  - Reproduction
    - **COMMIT** (persist) a container to a new image
    - **RUN** a new container from an image
  - Sleep
    - **KILL** a running container
  - Wake
    - **START** a stopped container
  - Death
    - **RM** (delete) a stopped container
- Extinction
  - **RMI** a container image (delete image)

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

file | 15 lines (11 sloc) | 0.475 kb

Open | Edit | Raw | Blame | History | Delete

```
FROM ubuntu:12.04

RUN apt-get update

# Make it easy to install PPA sources
RUN apt-get install -y python-software-properties

# Install Oracle's Java (Recommended for Hadoop)
# Auto-accept the license
RUN add-apt-repository -y ppa:webupd8team/java
RUN apt-get update
RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
RUN apt-get -y install oracle-java7-installer
ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
```

https://docs.docker.com/engine/installation/windows/

Thank you

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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
Apache VCL	Apache v2	Multi-platform (Apache/ PHP)	Portal, XML-RPC	VMware ESX, ESXi, Server	No	No	Yes	No	Yes	No	No
AppLogic	Proprietary	Linux	GUI, CLI	Xen	Global Volume Store (GVS)	No	Yes	Yes	No	Yes	Yes
Citrix Essentials	Proprietary	Windows	GUI, CLI, Portal, XML-RPC	XenServer, Hyper-V	Citrix Storage Link	No	Yes	Yes	No	Yes	Yes
Enomaly ECP	GPL v3	Linux	Portal, WS	Xen	No	Amazon EC2	Yes	No	No	No	No
Eucalyptus	BSD	Linux	EC2 WS, CLI	Xen, KVM	No	EC2	Yes	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
OpenNEbula	Apache v2	Linux	XML-RPC, CLI, Java	Xen, KVM	No	Amazon EC2, Elastic Hosts	Yes	Yes	Yes (via Haizea)	No	No
OpenPEX	GPL v2	Multiplatform (Java)	Portal, WS	XenServer	No	No	No	No	Yes	No	No
oVirt	GPL v2	Fedora Linux	Portal	KVM	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
Platform VMO	Proprietary	Linux, Windows	Portal	XenServer	No	No	Yes	Yes	No	Yes	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

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

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 a2.4 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.

Table 2.1 EC2 Standard Instance Types

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 EC2instance 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

Figure 3: Conceptual OpenStack Architecture

Scanned with CamScanner

Logical OpenStack Architecture

Figure 4: Logical OpenStack Architecture

Scanned with CamScanner

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

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=PLvvQ7qimTOkmJFGS_uYlOA423PlvVmxOg (Installation)
- https://youtu.be/wzVSGGg4fsY?list=PLvvQ7qimTOkmJFGS_uYlOA423PlvVmxOg (Instance Creation)
- https://youtu.be/G1ZY4RorBiw?list=PLvvQ7qimTOkmJFGS_uYlOA423PlvVmxOg (Instance Creation with Volume)
- https://youtu.be/QFkfSgjJddI?list=PLvvQ7qimTOkmJFGS_uYlOA423PlvVmxOg (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**, **JoyentAccelerator**, **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 Manag eability 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

Reference (complete information of Live Migration) http://www.diva-portal.org/smash/get/diva2:7...

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.

Refererences: 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)

BITS Pilani

Cloud Computing

SEWP ZG527

IaaS

Really, what is iaas???

heard of 3 models of Cloud Computing?

Yes, Yes, IaaS, PaaS and SaaS

Key concepts of IaaS

- Cloudbursting: The process of off-loading tasks to the cloud during times when the most compute resources are needed

- Multi-tenant computing

- Resource pooling: **Pooling** is a resource management term that refers to the grouping together of resources (compute(cpu), network(bandwidth), storage) for the purposes of ***maximizing advantage*** and/or ***minimizing risk*** to the users
- Hypervisor

Two primary facets that make IaaS special

Elasticity:

Wikipedia: “In **cloud** computing, **elasticity** is defined as the degree to which a system (or a particular **cloud** layer) autonomously adapts its capacity to workload over time”

OR simply put “Ability of a system to **expand** or **contract** its dedicated resources to meet the demand”

&

Virtualization

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/

BITS Pilani

Cloud Computing

SEWP ZG527

IaaS

Really, what is iaas???

Amazon Web Services

Amazon Web Services Cloud

- Provides highly reliable and scalable infrastructure for deploying web-scale solutions
- With minimal support and administration costs
- More flexibility than own infrastructure, either on-premise or at a datacenter facility

AWS infrastructure services

Examples

AWS – EC2, EBS, S3, LB

IaaS for you

*Thanks, I feel so "**Clouded**" now*

BITS Pilani

Cloud Computing

SEWP ZG527

heard of 3 models of Cloud Computing?

Yes, Yes, IaaS, PaaS and SaaS

Openstack overview

- OpenStack is a collection of open source technologies delivering a massively scalable cloud operating system.
- OpenStack cloud operating system controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

Openstack Components

OpenStack conceptual architecture

BITS Pilani

Cloud Computing

SEWP ZG527

Virtual Machine Provisioning and Manageability 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.
- 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

• Server provisioning is defining server's configuration based on the organization requirements, a H/W, and S/W component (processor, RAM, storage, networking, operating system, applications, etc.).

VMs can be provisioned by

- Manually installing an OS,
- Using a preconfigured VM template,
- Cloning an existing VM, or importing a physical server or a
- Server from another hosting platform.
- Physical servers can also be virtualized and provisioned using P2V (Physical to Virtual)

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.

BITS Pilani

Cloud Computing

SEWP ZG527

Virtual Machine Migration Services

Migration service -

The process of moving a virtual machine from one host server or storage location to another;

There are different techniques of VM migration-

- Hot/live migration,
- Cold/regular 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

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

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

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 Migration Demo

- Using Proxmox deployment tool

Cloud Computing

Module 4

(Linux containers, Dockers and Orchestration tools)

BITS Pilani

Agenda

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Difference Between Application Containers and System Containers

Application containers

System containers

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Difference Between System Containers and Virtual Machines

Virtual machines

System containers

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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)

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.cpu=2 –limits.memory=200MiB
6. lxc execute firstlimited –bash
7. cat /etc/*release

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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).**

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Benefits of Cloud Orchestration

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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)

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

1. Chef

- It is used in infrastructure automation and allow us to automate and control vast numbers of computers in an aut omated, reliable, and scalable manner.
- Chef translates system administration tasks into reusable definitions, known as cookbooks and recip es.
- Cookbooks are the configuration units that allow us to configure and perform specific tasks within Ch ef 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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Chef Architecture Cont...

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Chef Architecture Cont…

- **Pull Configuration:** Nodes will automatically update themselves with the config urations 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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 cen tralized 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 **D**omain **S**pecific **L**anguage (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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Puppet (Cont…)

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Puppet

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Puppet Functionalities

- **Defining distinct configurations** for each and every host, and continuously checking and confir ming 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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

How Puppet Works?

- Puppet is based on a **Pull deployment model,** where the agent nodes check in regularly after ever y **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 per forms 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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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…)

2. Agents send Facts to the Master

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 in stallation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.**).**

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Puppet Blocks

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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).

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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',
 }
}
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

3. OpenStack

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

4. Heat (Cont...)

- It enables orchestration engine to launch multiple composite cloud app lications based on templates in the form of text files that can be treate d like code.
- A native Heat template format is evolving, but Heat also endeavours to provide compatibility with the AWS CloudFormation template format, s o 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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

How heat works?

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 com pletely launch your application.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 that 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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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)

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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)**.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

basic-stack.yaml
(Note: A Stack means a group of connected cloud resources)

```yaml
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 en
vironment 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
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

basic-stack.yaml (cont….)

```yaml
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
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

basic-stack.yaml (cont….)

```yaml
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 }
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

basic-stack.yaml (cont….)

```
      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 ]}
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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                 |
+---------------------+--------------------------------------+
```


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Application modelling?

- In modern environments, Even simple applications may require several other applications in order to function - li ke 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'.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Application scenario with Charms and Bundles

- Modern applications are typically composed of many applications - databas es, 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 an d their relationships - what is connected to what - and we use **a bundle for** t hat.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 dep loyment in seconds with the right bundle.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.**

BITS Pilani, Deemed to be University under Section 3, UGC Act

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/

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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 predeter mined 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.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.**

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

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.			

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956