



**BITS Pilani**  
Pilani Campus

# BITS Pilani presentation

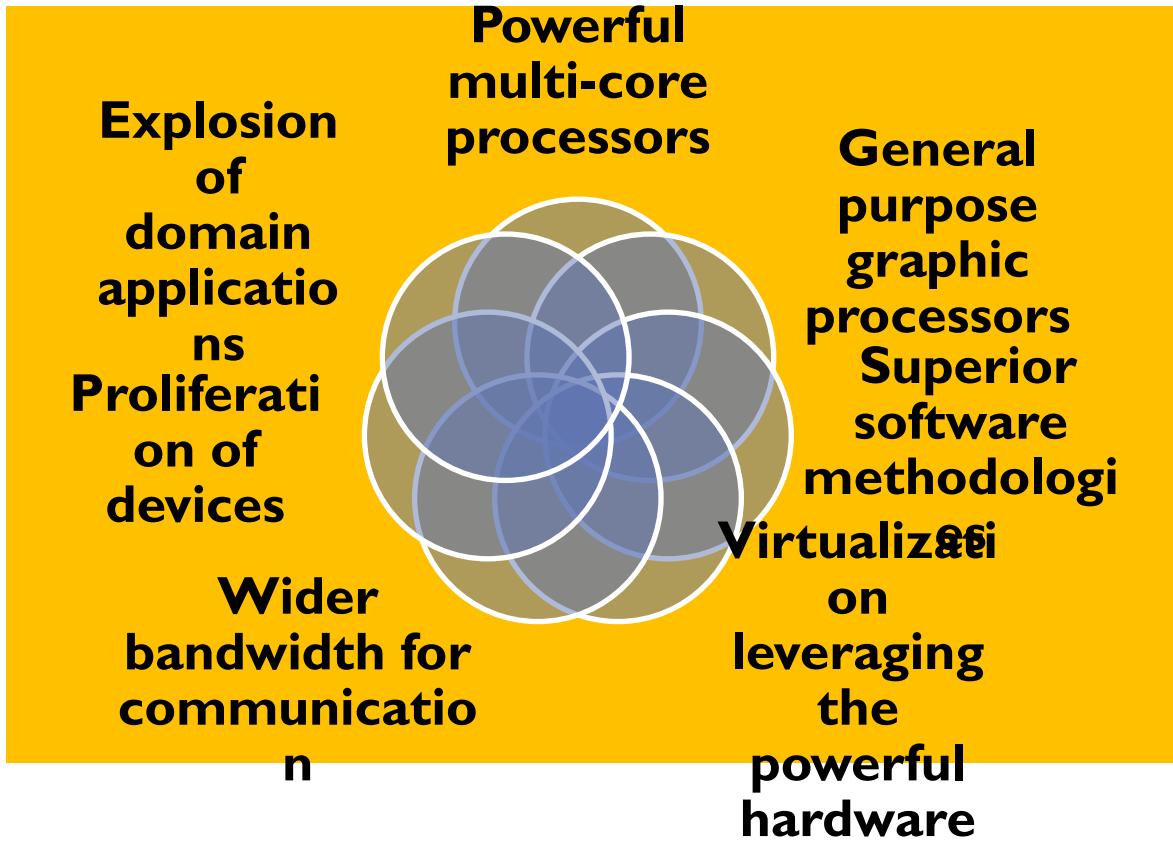
Mridul Moitra  
Cloud Computing





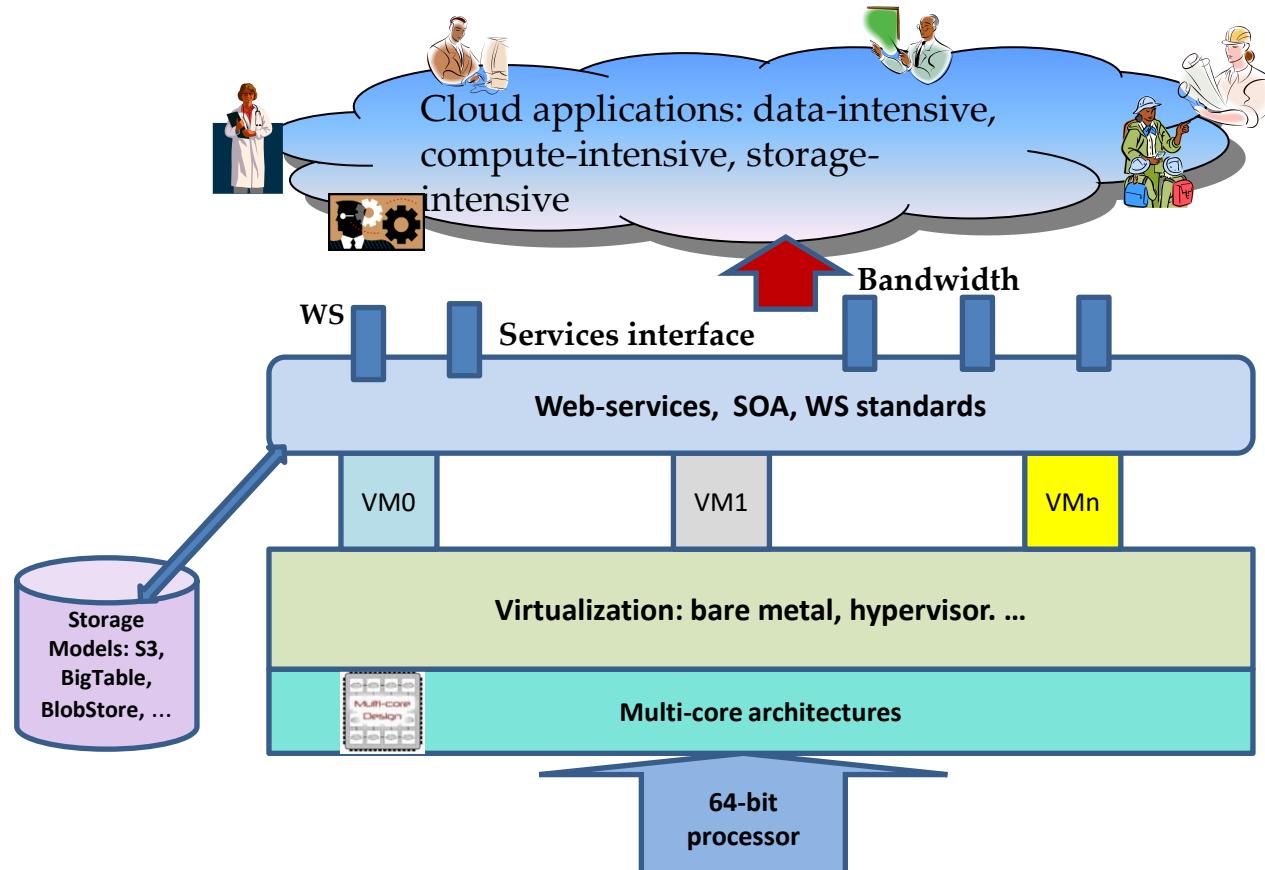
# **<CSI ZG527 / SS ZG527 / SE ZG527 Cloud Computing Lecture No. 1**

# Motivation



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

# Technology Advances



# Evolution of cloud computing

?

- **The evolution of cloud computing can be bifurcated into three basic phases:**
- **1. The Idea Phase-** This phase inceptioned in the early 1960s with the emergence of utility and grid computing and lasted till pre-internet bubble era. Joseph Carl Robnett Licklider was the founder of cloud computing.
- **2. The Pre-cloud Phase-** The pre-cloud phase originated in 1999 and extended to 2006. In this phase the internet as the mechanism to provide Application as Service.
- **3. The Cloud Phase-** The much talked about real cloud phase started in the year 2007 when the classification of IaaS, PaaS, and SaaS got formalized. The history of cloud computing has witnessed some very interesting breakthroughs launched by some of the leading computer/web organizations of the world.

# 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 up 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....

---



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
- The hardware and software services are available to
- general public, enterprises, corporations and businesses markets

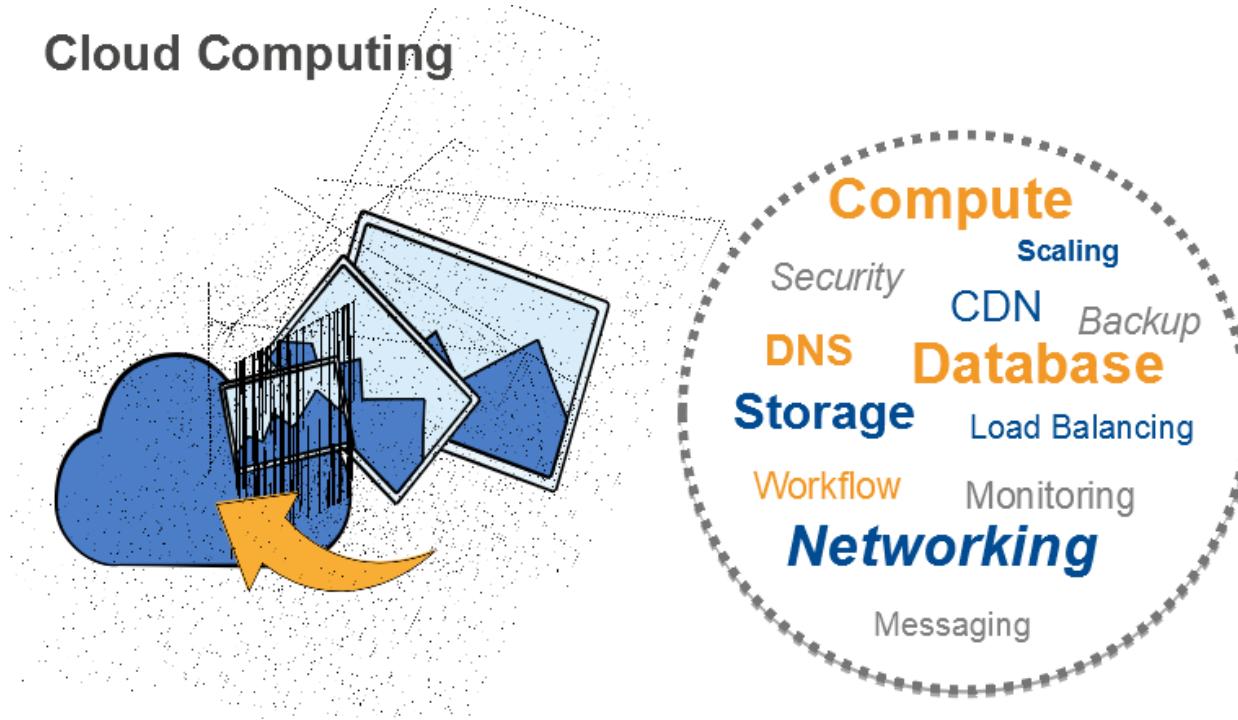
# Cloud Computing: Definition

---

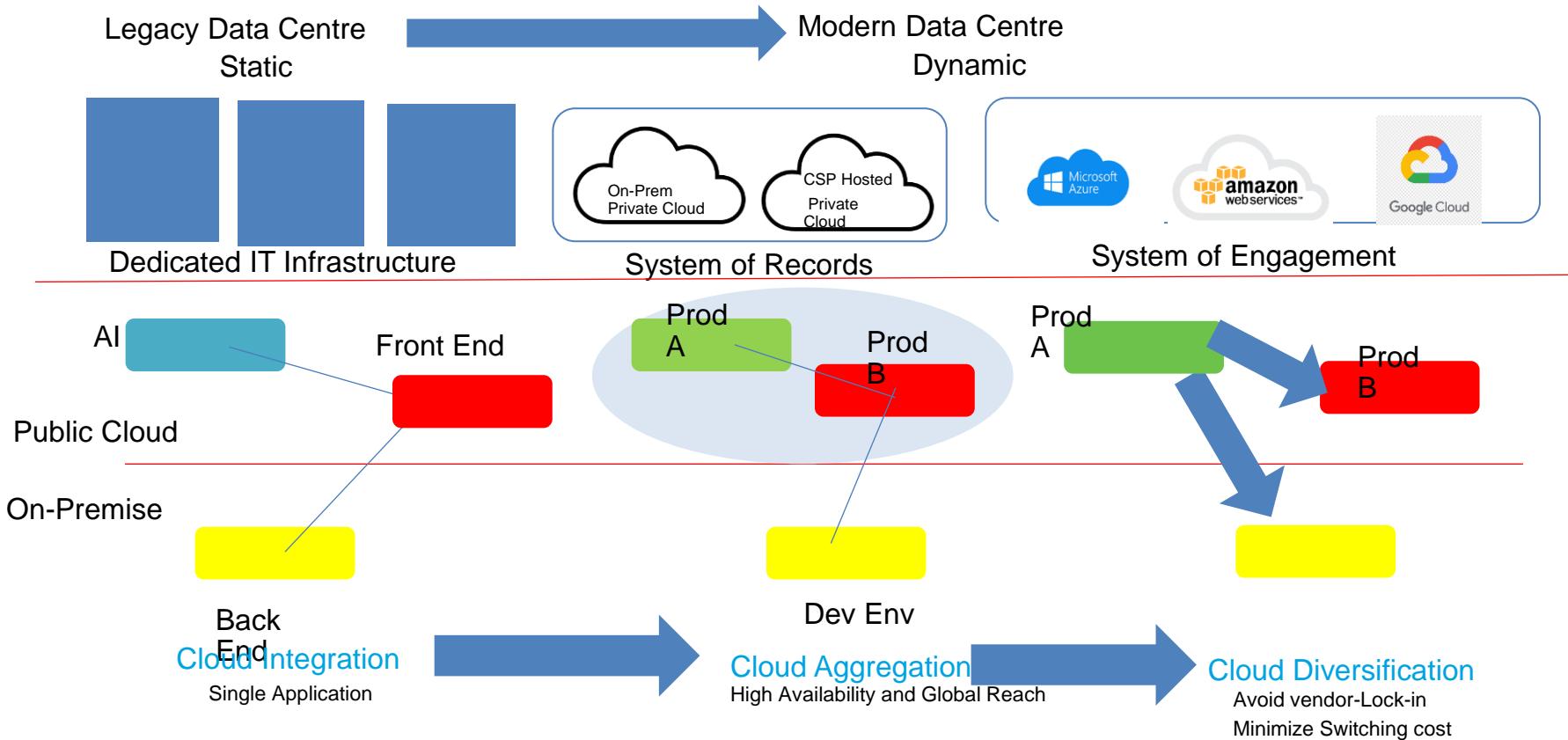
The US National Institute of Standards (NIST) defines cloud computing as follows:

*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



# Cloud Transformation Journey



# Challenges Of the CIO



# Drivers for the new Platform

## Generational Shift of Computing Platform

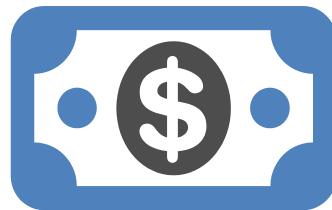
Technology	Economic	Business	
	Centralized compute & storage, thin clients	Optimized for efficiency due to high cost	High upfront costs for hardware and software
	PCs and servers for distributed compute, storage, etc.	Optimized for agility due to low cost	Perpetual license for OS and application software
	Large DCs, commodity HW, scale-out, devices	Order of magnitude better efficiency and agility	Pay as you go, and only for what you use

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

# Cloud Computing Business Drivers



## Cost optimisation



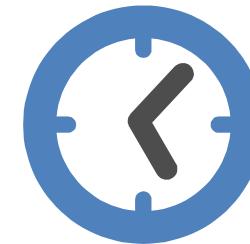
- No capex, less assets
- Pay-as-you-use
- On-demand capacity
- Elasticity
- Economies of scale
- Time-to-value

## Risk optimisation



- Business continuity
- Technology independence
- Operational complexity
- Specialised skills

## Strategic agility



- Time-to-market
- Innovation
- New business models
- Resource leverage
- Adaptability
- Flexibility

...why would one not consider these benefits?

# 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

# Cloud Summary



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

# Characteristics of Cloud Computing



## 5 Essential Characteristics of Cloud Computing

Ref: The NIST Definition of Cloud Computing

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



On-demand  
self-service



Ubiquitous  
network  
access



Location  
transparent  
resource  
pooling



Rapid  
elasticity



Measured  
service with  
pay per use

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

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

## Essential Characteristics

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

## Service Models

- Software as a Service
- Platform as a Service
- Infrastructure as a Service

## Deployment Models

- Private
- Public
- Hybrid
- Community

## Traditional Infrastructure

## Amazon Web Services



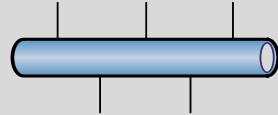
Security



Security Groups

NACLs

Access Mgmt



Networking



“Public”  
EC2 “Classic”

ELB

VPC



Servers



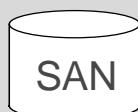
AMI

Servers

Amazon EC2



Instances



RDBMS

Storage &  
Database



Glacier



EBS



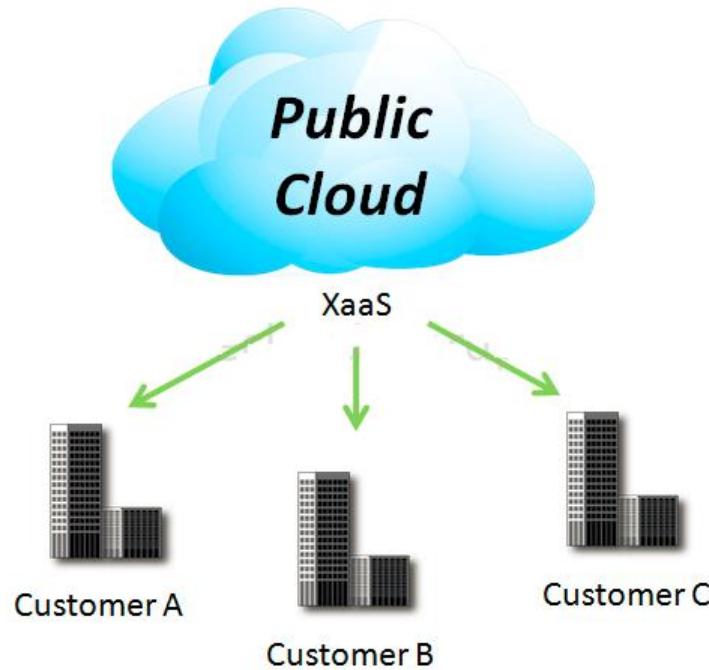
S3



RDS

# 4 Deployment Models

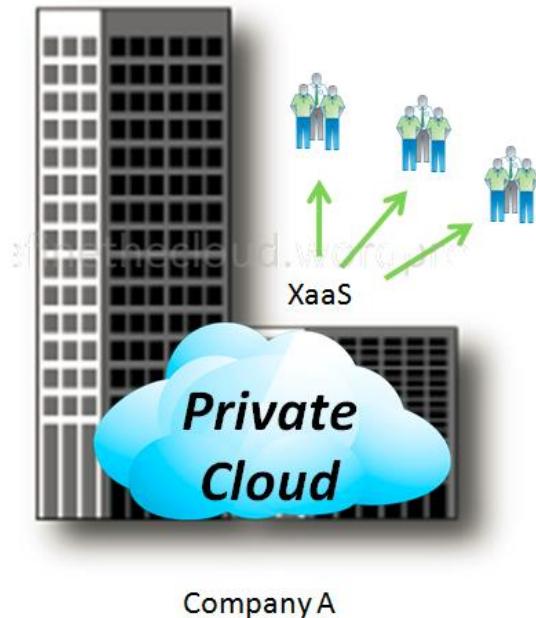
## 1. Public Cloud



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

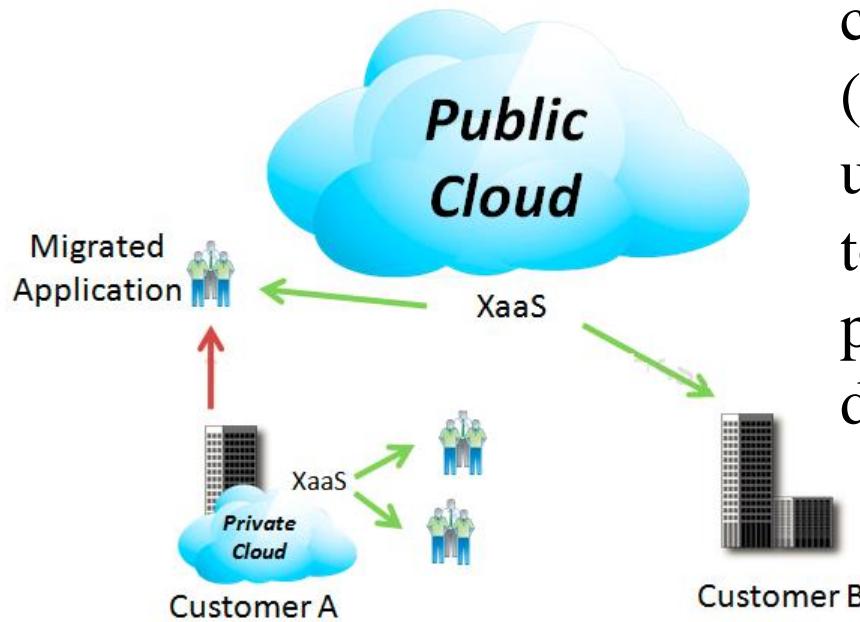
## 2. Private Cloud



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

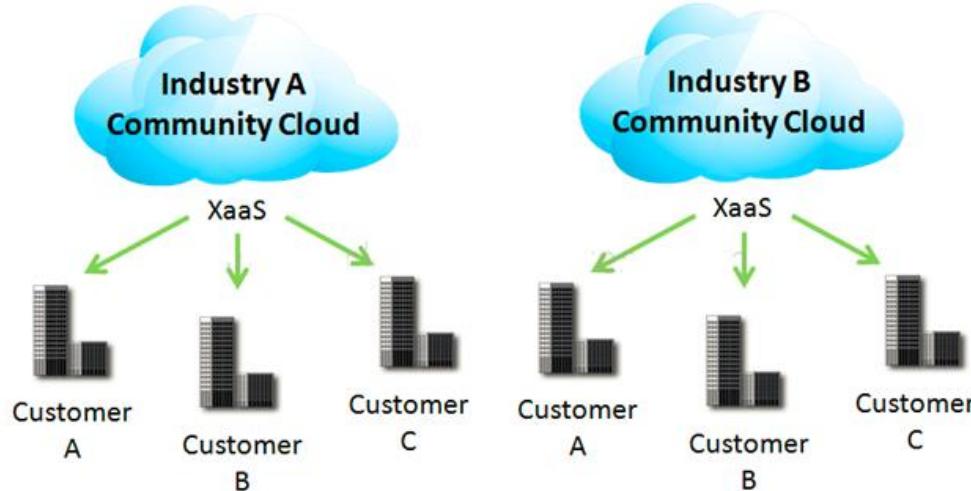
## 3. Hybrid Cloud



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

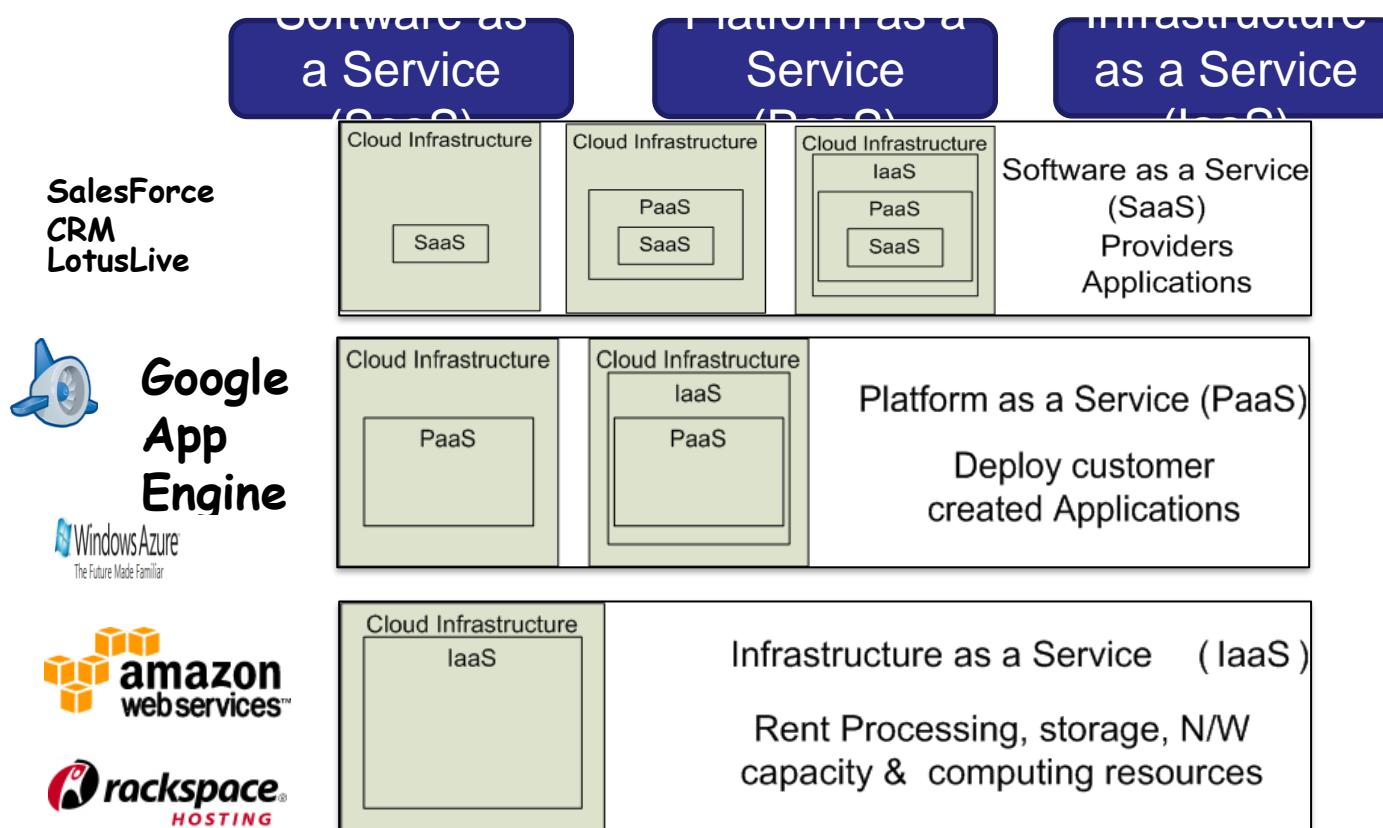
# 4 Deployment Models

## 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

# 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.*

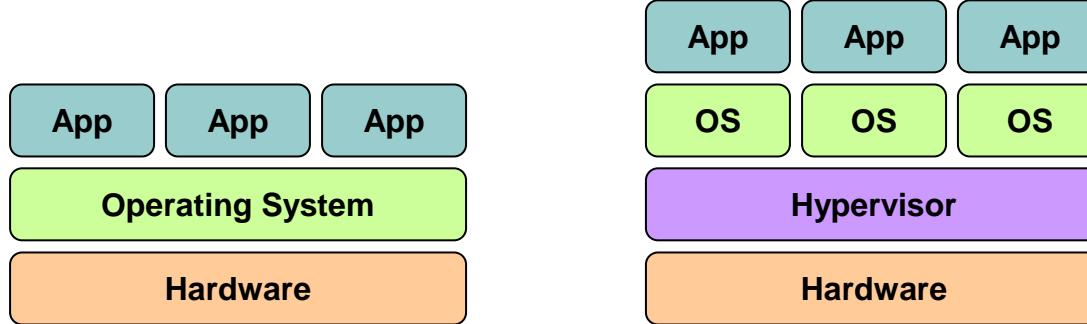
# Infrastructure as a Service



**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.**

## 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



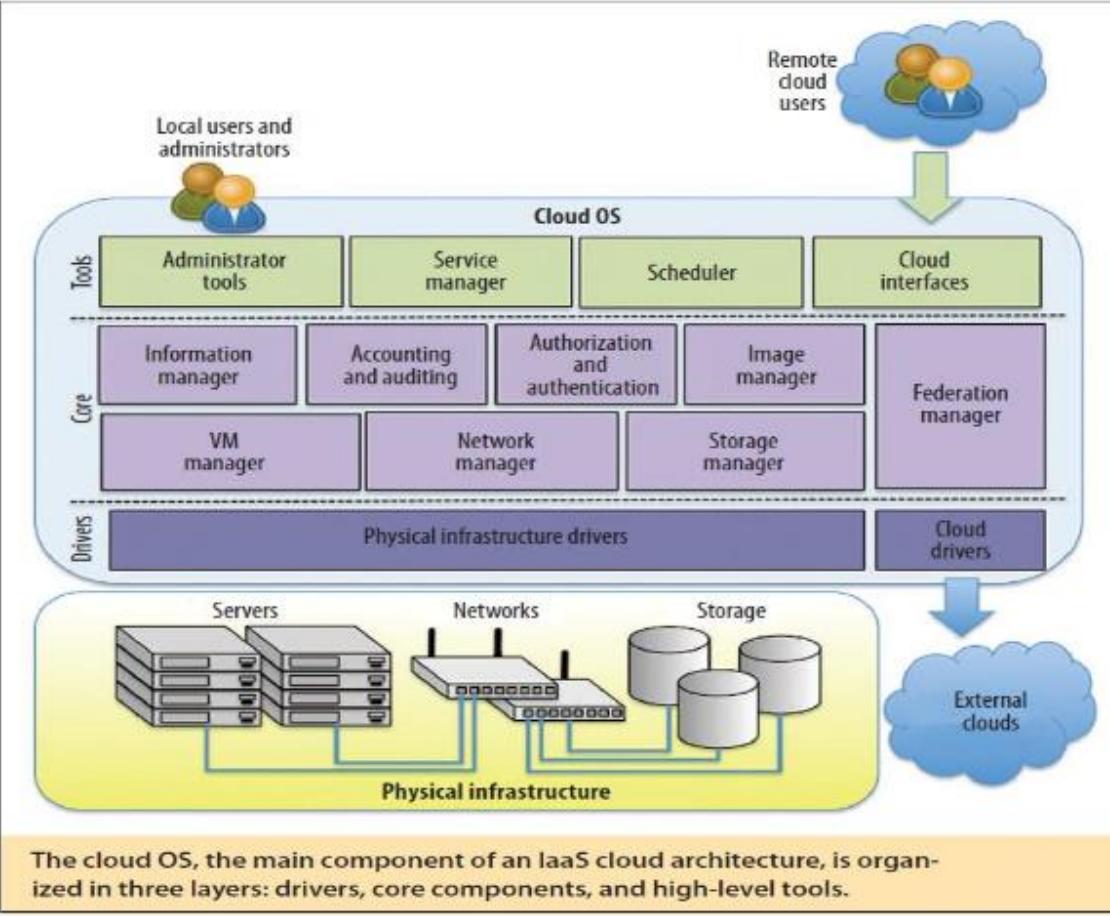
- Provide on-demand provisioning of computational resources
- Use virtualization technologies to lease these resources
- Provide public and simple remote interfaces to manage those resources
- Use a pay-as-you-go cost model, typically charging by the hour
- Operate data centers large enough to provide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

## Distributed Management of Virtual Machines

## Reservation-Based Provisioning of Virtualized Resources

## Provisioning to Meet SLA Commitments

# The Cloud OS



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

- Why is it becoming a Big Deal:
  - Using high-scale/low-cost providers,
  - Any time/place access via web browser,
  - Rapid scalability; incremental cost and load sharing,
  - Can forget need to focus on local IT.
- Concerns:
  - Performance, reliability, and SLAs,
  - Control of data, and service parameters,
  - Application features and choices,
  - Interaction between Cloud providers,
  - No standard API – mix of SOAP and REST!
  - Privacy, security, compliance, trust...



Implementation of cloud services on resources that are shared between many customers, managed off-premises



Typically, cloud provider owns and controls the resources/assets, definition of services, costs and risks

Variations exist – such as hosters and integrated cloud platforms



Cloud solutions

Office 365 (SaaS)  
Dynamics CRM Online (SaaS)  
EC2, RDS  
Windows Azure platform (PaaS)



Implementation of cloud services on resources that are dedicated to your organization, whether they exist on-premises or off-premises



Typically, your organization owns and controls the resources/assets, definition of services, costs and risks

Variations exist – such as hosted and virtual private clouds



Cloud solutions

Windows Server 2008 R2 Hyper-V,  
System Center (IaaS)  
Windows Azure Appliance (PaaS)  
Database Services

## Private Cloud

---

- **Private clouds are cloud infrastructures that are deployed for a single organization.**
- **These can be managed internally or externally, but all systems and infrastructure are for the purposes of the organization.**
- **When considering a private cloud, the biggest decision that a business needs to make is the scope of the needed investment to create the private cloud, as implementation can be very expensive.**

## Cloud vs. Public Cloud

---

- More than a location and ownership distinction
- ▶ Private Cloud
  - ▶ Control
  - ▶ Conventional storage
  - ▶ Custom policies
  - ▶ Heterogeneous infrastructure
  - ▶ Regulatory compliance & data sovereignty
- ▶ Public Cloud
  - ▶ Scale
  - ▶ Cloud storage
  - ▶ Common policies
  - ▶ Homogeneous infrastructure
  - ▶ Work in progress

Mixed/blended model of private and public clouds

- Variations and multiple interpretations exist

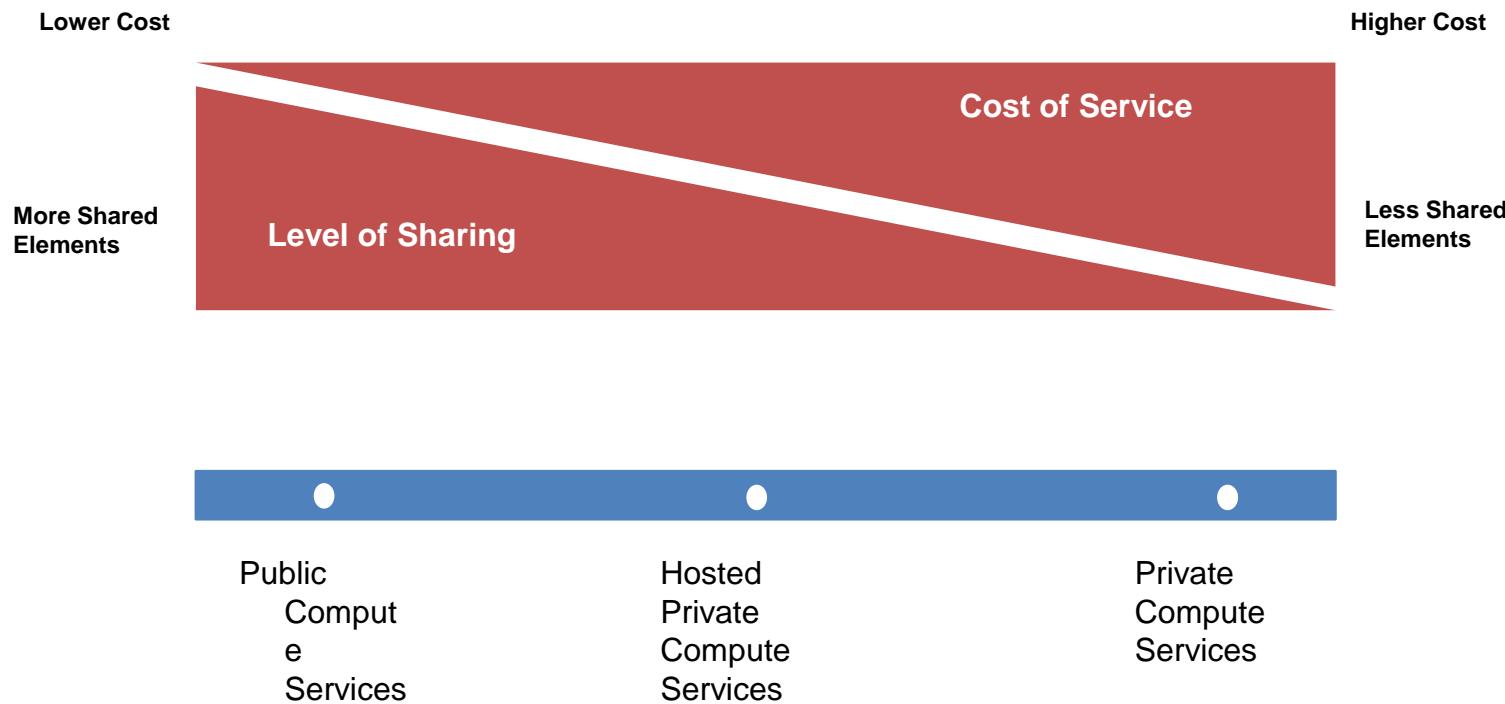
On-premises and off-premises bridging

- Most common scenario today
- Especially for large enterprises

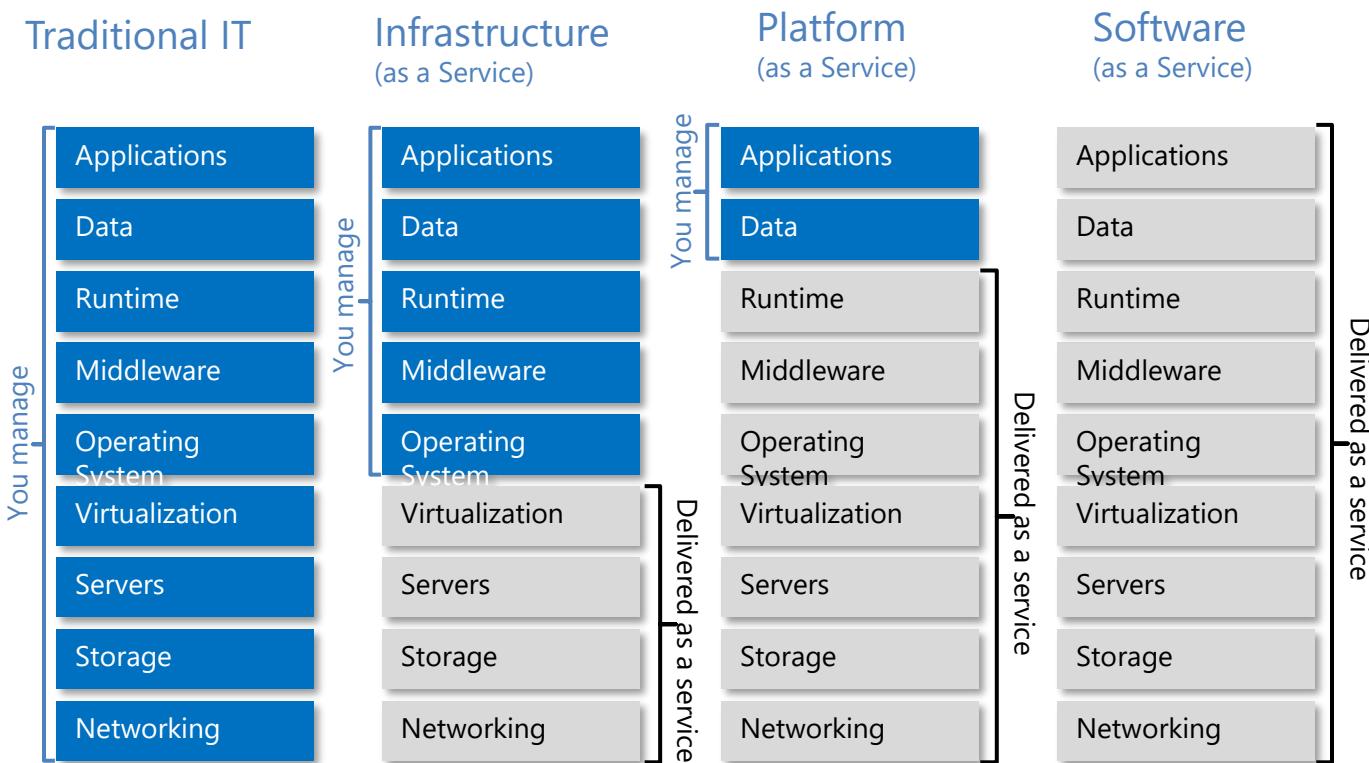
More than a deployment / delivery model

- Application design, architectural model

# Product Families and Cost Principles



# Cloud Service Models



## Cloud providers



**Microsoft**



**at&t**

**Cloud  
Solutions**



SOME RIGHTS RESERVED

 **rackspace**  
the open cloud company



**iCloud**



**CITRIX®**

 **hp** Cloud Services

 **amazon  
web services™**

 **bluelock.**  
IB



 **IBM Cloud**

 **Google**  
App Engine

 **Joyent**

 **RedMonk**

Partial list of cloud companies

---

Businesses that are best suited for using the public cloud are ones that need to bring a product/service to market quickly without the internal infrastructure and support to build out their own private cloud system. It does provide smaller companies without IT departments the opportunity to match the deployment speed of larger companies.

Increased network efficiency and more resources

Reduced complexity and lead times (because the architecture is fixed)

Ready-to-go applications used within the public cloud can conform to the demands of business

Disadvantages Of Public Cloud:

Fewer options for customization

Substantially less secure

Fixed architecture cannot (at times) grow with the needs of the business

- Businesses that are best suited for the private cloud are ones that must comply with
- regulatory guidelines or have highly volatile applications needed within the cloud.
- Additionally, these businesses will be required to install their own servers and storage
- hardware that allows for modifications in workloads. Though this can be a significant
- investment, it is required if the business deals with regulated data or must comply with
- industry rules.

## Advantages Of Private Cloud:

Extensive security options and capabilities, substantially more than the public cloud availability to the internal network and increased access/communication for internal users can grow with a business and be expanded or changed as needed

## Disadvantages Of Private Cloud:

Significant level of engagement from both management and IT departments are required

Significant investment is required, both to build the private cloud and maintain/grow it  
Does not deliver the short-term solutions – given the required time needed to build it out – that the public cloud does

## Advantages Of Hybrid Cloud:

Best of both the public and private cloud in terms of needed resources

Added accessibility for internal and external users

Security parameters are higher than that of the public cloud

## Disadvantages Of Hybrid Cloud:

Inherent inefficiency of monitoring several different security platforms

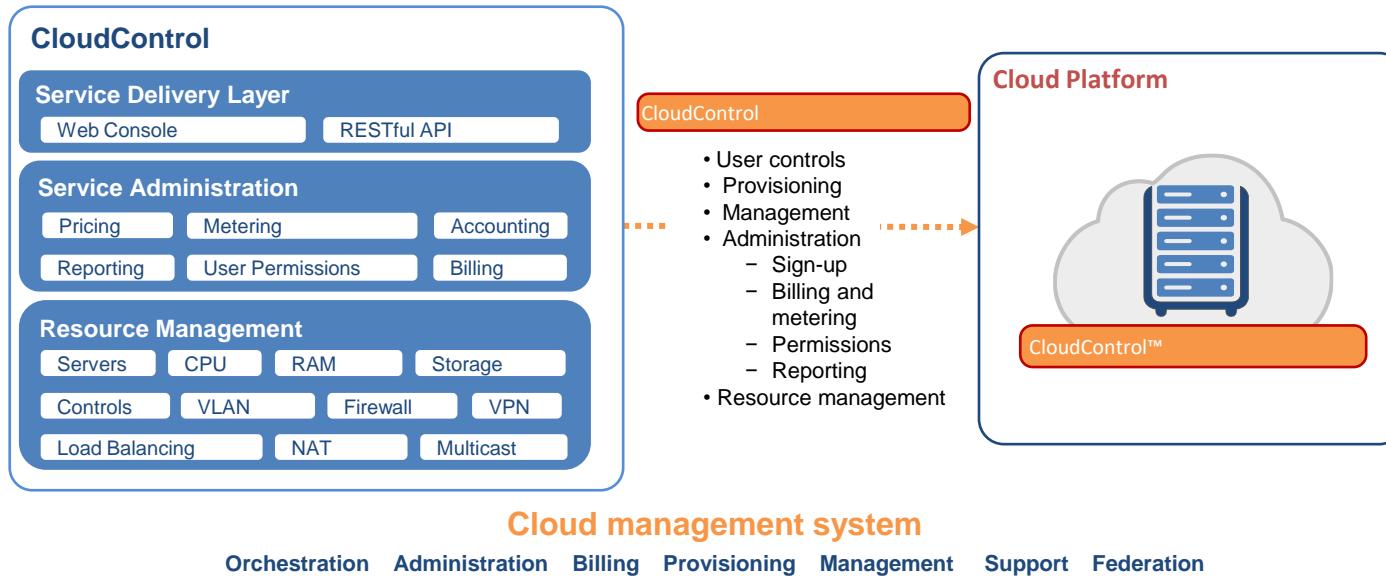
Customization of rules and policies to govern security and the elements of infrastructure that do not link between the public and private cloud

Added security risk

Businesses that are best suited to the hybrid cloud are those with sites that have unreliable fluctuations in traffic. With e-commerce, these fluctuations vary greatly between times of day and seasons in the year. Because the hybrid cloud has the same security of the private cloud, businesses can utilize both the added resources and the security to facilitate the needs of their customers. In essence, it is the best of both worlds.

# Cloud management system

Addresses the complexity of cloud orchestration, provisioning and billing



# Consumption Model



Model	<u>Open Community</u>	<u>Controlled Open Mode</u>	<u>Contractual Open</u>	<u>Public/Private Hybrid</u>	<u>Private Closed</u>
Examples	Facebook Twitter LinkedIn MyFitnessPal Google Groups	IBM SmartCloud Enterprise Amazon Web Services RackSpace OpSour	Salesforce.com Workday MailChimp QuickBooks Online	IBM SmartCloud HP Cloud Service Microsoft Azure	Internal but can be implemented by a third-party vendor
Characteristics	No SLA  No Contract	Simple SLA  Transactional pricing	SLA with no indemnification  Contract	SLA guaranteeing uptime  Contract	Explicit SLA  Capital expense with ongoing maintenance
	Simple Password Protection  No governance model	More security  No explicit governance	High security provided  Governance in place	Highest level of security  Explicit governance	Secure platform  Explicit governance

- Elastic/burst capacity (e.g., apps with variable load, HPC / parallel processing, etc.)
- Temporal applications (e.g., marketing apps, test & QA environments, etc.)
- Cloud-based DMZ / Perimeter Zone
- High Performance Compute
- Backup and storage
- Disaster recovery

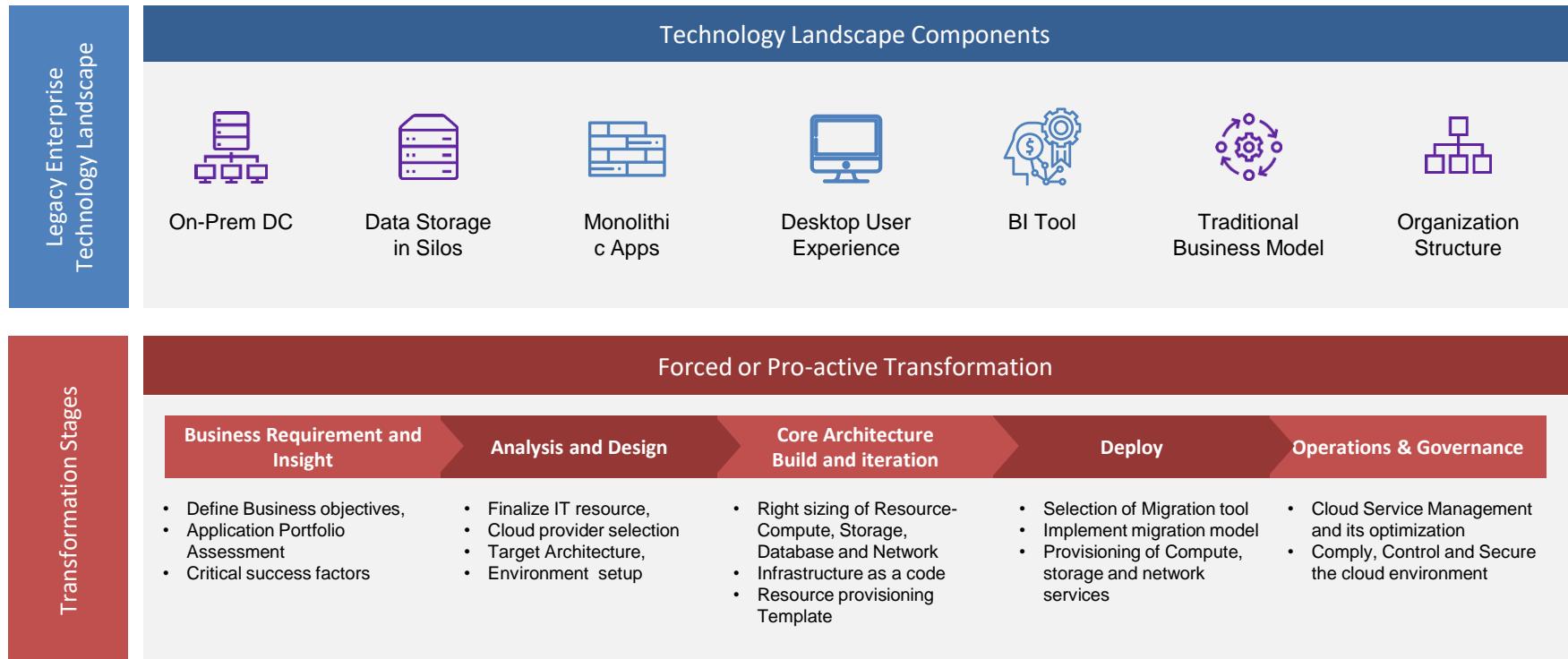
These work, but they are still deployment models



## Topics of Interest as GF

TOPIC DESCRIPTION		
DSE*ZG522	Big Data Systems	What is big data - are existing systems sufficient?; Data Warehouse v/s Data Lakes; Hadoop – Components; Storage - Relational DBs/ NoSQL dbs / HDFS / HBase / Object Data stores - S3; Serialization; Interfaces - Hive/ Pig; Stream Processing; Spark; Mahout.
SS ZG527	Cloud Computing	Concurrency and distributed computing, message passing over the network, connectivity and failure models, local vs remote connectivity, distributed resource modeling, distributed data models; replication & consistency; virtualization; CPU virtualization, memory and storage virtualization, virtualized networks, computing over WAN and Internet; computing on the cloud, computing models, service models and service contracts, programming on the cloud; Cloud infrastructure, LAN vs Wan issue, resource scaling and resource provisions, performance models, scalability, performance measurement and enhancement techniques; cloud applications and infrastructure services.
IS ZC446	Data Storage Technologies & Networks	Storage Media and Technologies – Magnetic, Optical and Semiconductor media, techniques for read/write operations, issues and limitations. Usage and Access – Positioning in the memory hierarchy, Hardware and Software Design for access, Performance issues. Large Storages – Hard Disks, Networked Attached Storage, Scalability issues, Networking issues. Storage Architecture. - Storage Partitioning, Storage System Design, Caching, Legacy Systems. Storage Area Networks – Hardware and Software Components, Storage Clusters/Grids. Storage QoS – Performance, Reliability, and Security issues.
SS ZG515/DSE*ZG515	Data Warehousing	Introduction, evolution of data warehousing; decision support systems; goals, benefit, and challenges of data warehousing; architecture; data warehouse information flows; software and hardware requirements; approaches to data warehouse design; creating and maintaining a data warehouse; Online Analytical Processing (OLAP) and multi-dimensional data, multi-dimensional modeling; view materialization; data marts; data warehouse metadata; data mining.
CSI** ZG522	Design and Operation of Data Centers	Data Center Design: Principles (Scalability, Reliability, and Elasticity), Components - Computing Infrastructure (Processing, Storage, and Networking) and Physical Infrastructure (Power, Cooling, and Physical Security); Servers – Server Hardening, Server Optimization, Server Deployment and Consolidation, Converged and Hyper-Converged Infrastructure. Application monitoring and maintenance. Networking for data centers – device hardening, bandwidth aggregation, traffic management, redundancy, network isolation, deployment of internal security and peripheral security; Contingency Planning & Disaster Recovery: Backup, recovery, and redundancy/replication technologies and approaches. Data Center Architecture: Private, Public, and Hybrid models; Distributed Data Centers; Introduction to Software Defined DataCenters. Costing and Pricing– Costing and Cost Optimization, Pricing and Economics of Data Center Operation.
	Ethics for Data Science	Introduction to data ethics, perils of big data, foundations of data privacy, challenges of privacy in the digital age, data policies, consent and fair usage.
	Infrastructure Management	Introduction to System Management IT Infrastructure, Introduction to System Management IT Infrastructure, Staffing Legislation , Ethics, Outsourcing for ITSM, Customer Service, Availability, 6. Performance and Tuning, Production Acceptance, Change Management, Problem Management, Storage Management, Network Management, Configuration Management, Capacity Planning, Strategic Security, Business Continuity, Facilities Management, Developing Robust Processes, Using Technology to Automate and Evaluate Robust Processes, Integrating Systems Management Processes, Special Considerations for Client-Server and Web-Enabled Environments
DSE*ZG523	Introduction to Data Science	Context and use of Data Science. Highdimensional data, graphs, vectors in high dimensional space and large matrices; Algorithms for massive data problems, sampling techniques. Techniques for extracting information/patterns from data.
CSI** ZG515	Introduction to DevOps	Continual Service - continuous integration and continuous delivery; Scaling: automating infrastructure and infrastructure-as-code; DevOps and Cloud: platform-as-a-service and DevOps, use of virtual machines and containers for deployment, Micro-services; application lifecycle management: deployment pipeline and application deployment,continuous deployment pipeline; stack management - life cycle of stack and events, resource and event monitoring, auto healing; Security: security of deployment pipeline, policy-as- code.
CSI** ZG511	IT Infrastructure Projects & Processes	The course introduces infrastructure software project process as a structured methodology for professional software Maintenance and infrastructure. Introduction to Software process, Introduction to ITIL, CMMI – SVC – Capacity Maturity Model integrated for Services.

# Cloud Adoption Journey – Enterprise Landscape



# Key Cloud Adoption Drivers

 Global Scale

**Worldwide reach, Hyperscale & Elasticity** with **economic** benefits

 Digital User Experience

**Omni channel** experience on **Mobile** platforms & enabling user **Mobility**

 Reliability & Resiliency

**Reliable trading - High Availability**, Dynamic Failover, **COVID** lessons learnt

 Compliance & Security

**Regularity** Compliance, **Data Protection** & **confidentiality**

 IT Simplification

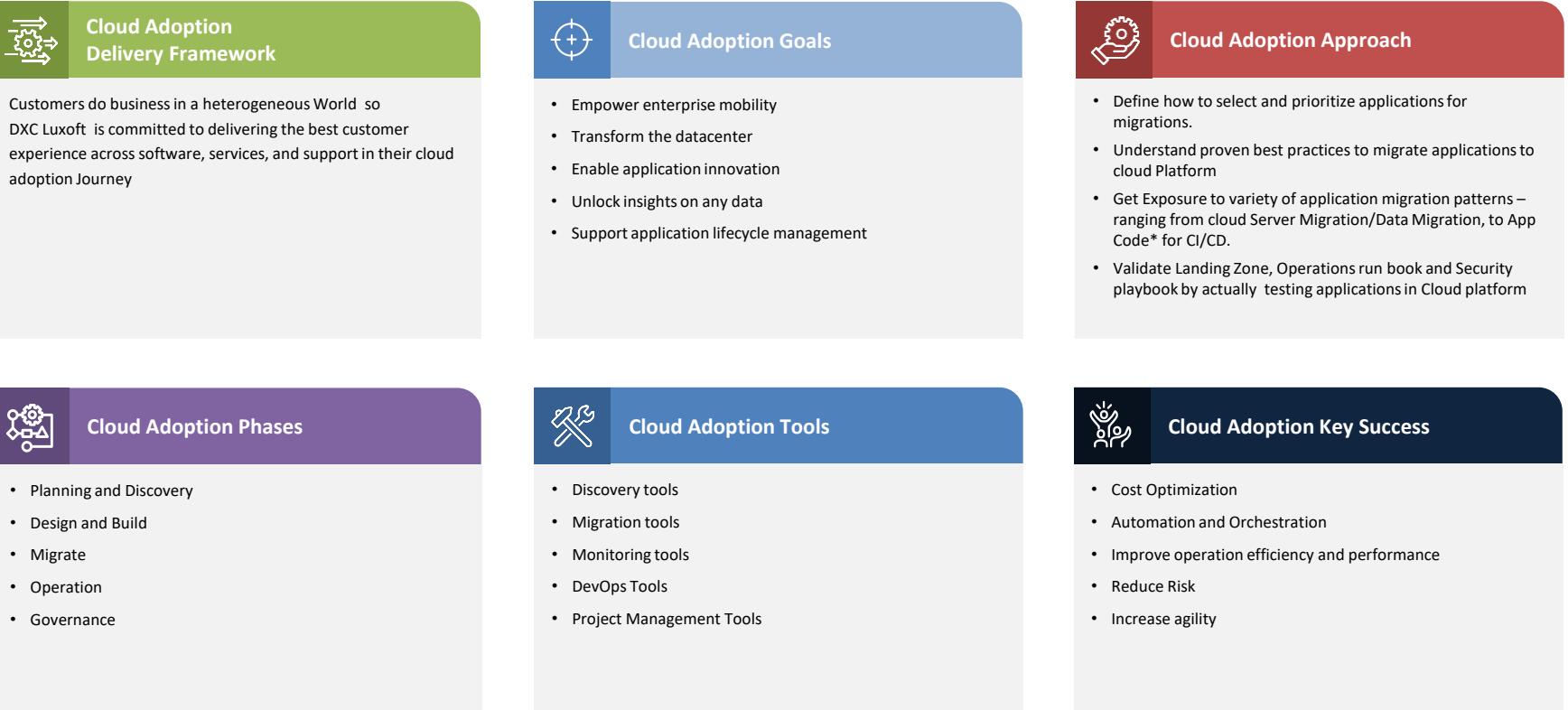
**Accelerated Time to Market - DevOps, DevSecOps, Managed PaaS**

 Driving Innovation

**SaaS** leverage - **Microservices, Analytics, AI, Blockchain and IoT** enabling business workflows



# Cloud Adoption Journey Road Map



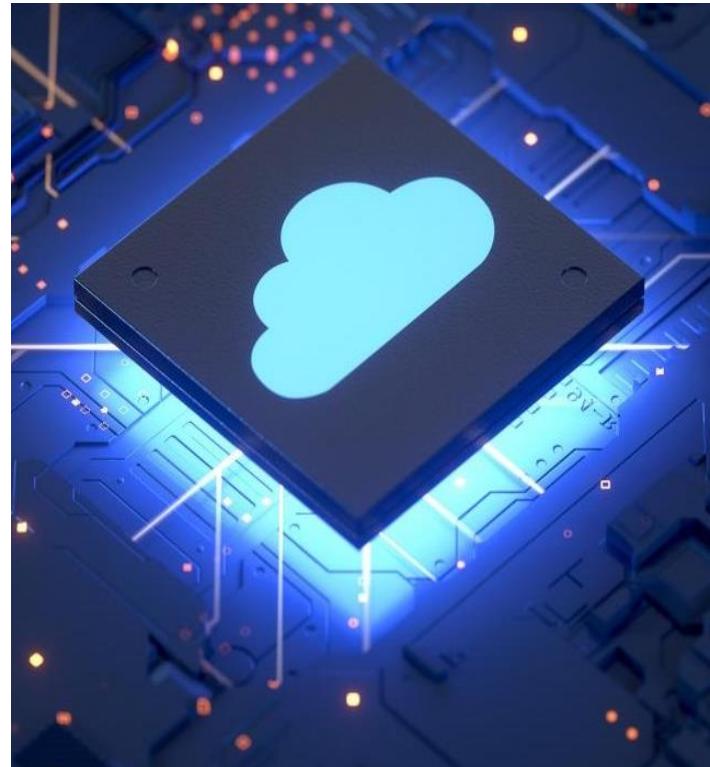
# Cloud Adoption

## Framework (CAF)

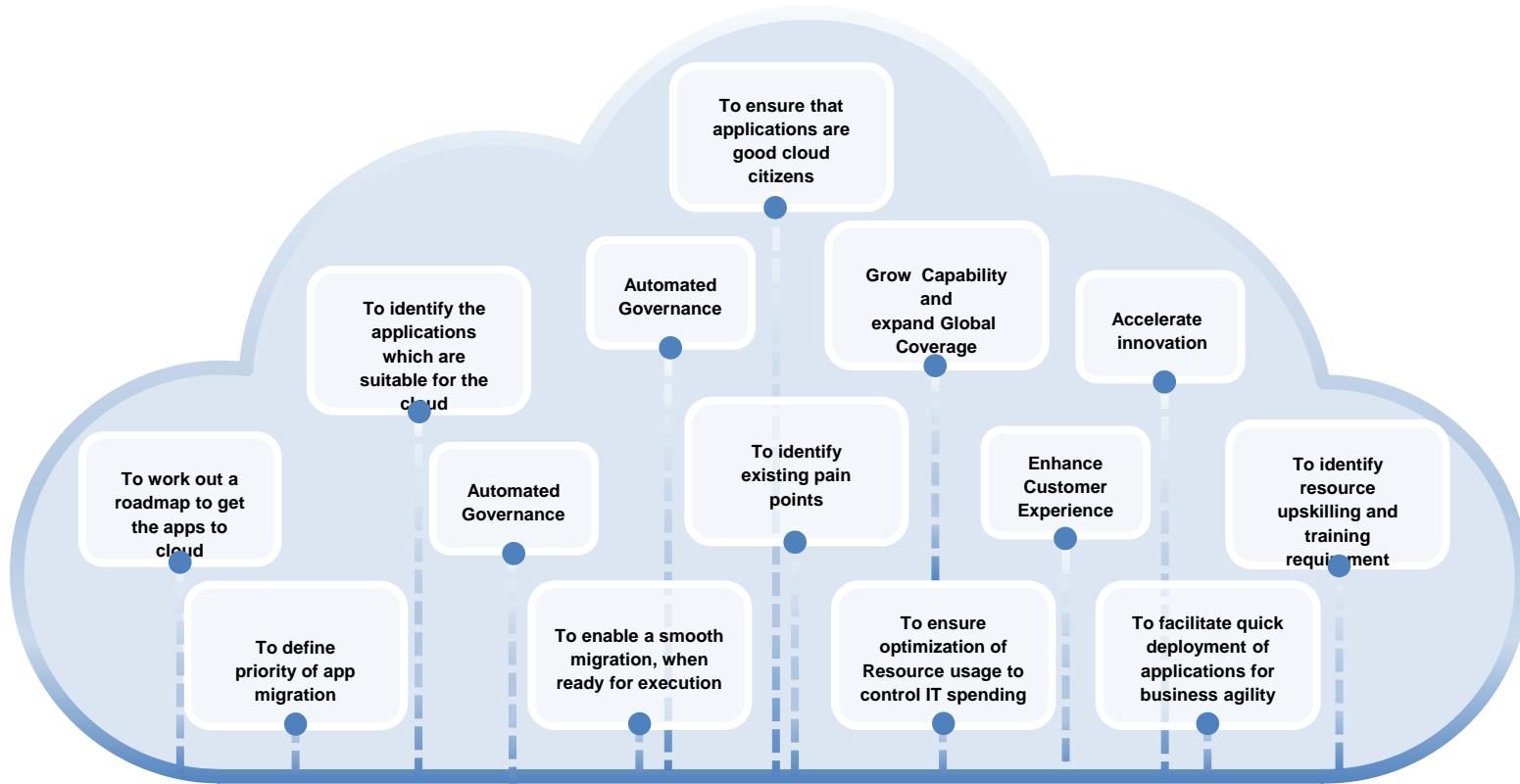
The Cloud Adoption Framework brings together cloud adoption best practices. It provides a set of tools, guidance, and narratives that would help shape technology, infrastructure, and people strategies for driving your desired business outcomes during your cloud adoption effort.

### The cloud adoption framework objectives includes:

- The Cloud Adoption Framework provides tools and guidance for implementing cloud technologies to incorporate business, people and process changes,
- Cloud Adoption framework" is used to describe collections of development tools to middleware to database services that ease the creation, deployment and management of cloud applications
- Aligns cloud adoption with business objectives across the cloud adoption stages.
- CAF Standardize technology adoption to reduce technology debt and streamlines cloud services management.
- CAF ensures security of infrastructure, applications, and data, while ensuring data sovereignty.
- CAF drives audit readiness for infrastructure applications.
- CAF allows periodical review of the reference architecture, approved list of services, security controls and cost optimization techniques.



# Why do we need a Cloud Adoption Framework



---

In [software engineering](#), **SOA (service-oriented architecture)** is an architectural style that focus on discrete services instead of a monolithic design.<sup>[1]</sup> By consequence, it is as well applied in the field of [software design](#) where services are provided to the other components by [application components](#), through a [communication protocol](#) over a network. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online. SOA is also intended to be independent of vendors, products and technologies.

A **web service** is any piece of software that makes itself available over the internet and uses a standardized XML messaging system



**BITS** Pilani  
Pilani Campus

# BITS Pilani presentation

Mridul Moitra  
Cloud Computing





**BITS** Pilani

<CSI ZG527 / SS ZG527 / SE ZG527  
Cloud Computing  
Lecture No. 2

# Overview

- **Introduction to Virtualization and Hypervisors**
  - Origins of Hypervisors
  - Hypervisors and its Classification
  - Existing Solutions/Products
  - Advantages and Disadvantages
  - State of the Art
  - Future of Hypervisors and Virtualization

# Introduction

Virtualization is the simulation of the software and/or

- hardware upon which other software runs. This
- simulated environment is called virtual machine. Each
- VM can run its own operating systems and
- applications as if it were in a physical machine. So It is
- way to run multiple operating systems on the same
- hardware at the same time.
- ☐ For e.g., Windows and Linux both can run on the same
- laptop at the same time

# Origin of Hypervisor

- IBM first developed CP/CMS operating system in 1967, an attempt to build time-sharing systems for mainframe systems
- In 1972, IBM's zSeries line featured Virtualization
- Early acceptance and rapid development by developers all over
- In 1985, IBM introduced the PR/SM hypervisor to manage logical partitions
- Other companies, Sun Microsystems, HP, and SGI joined the race and started selling virtualized software around 2000

# What is Virtualization

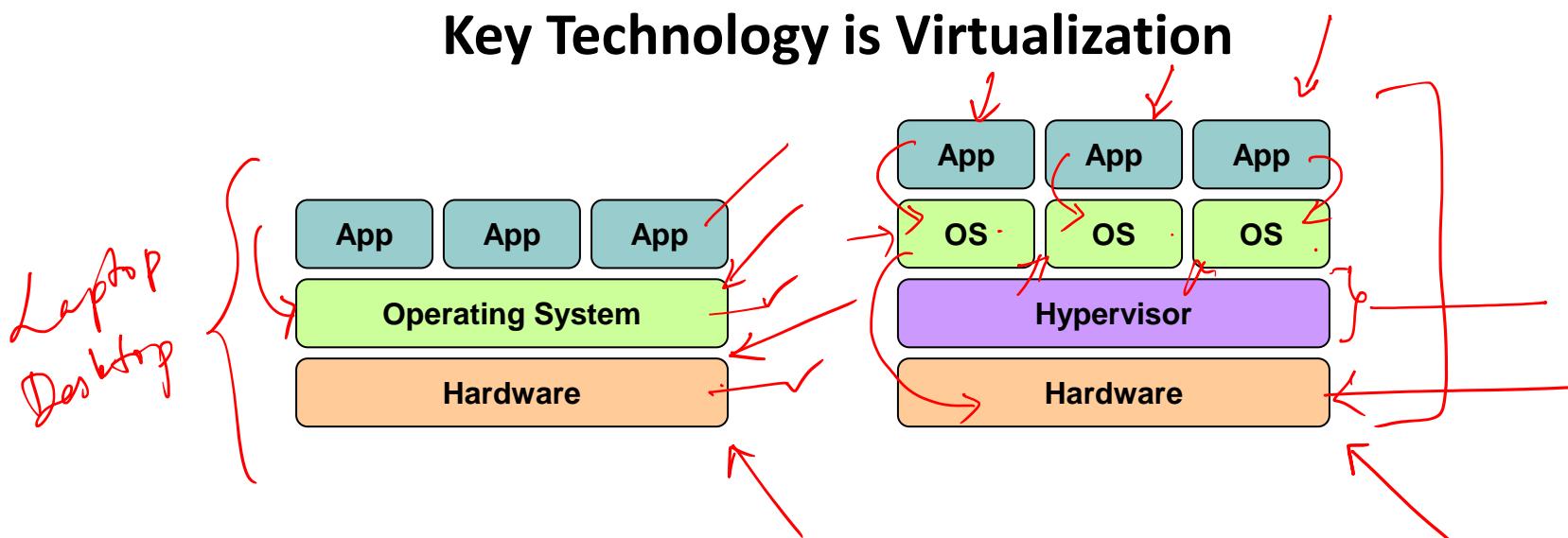
- Before using Virtualization, we had:
  - Single OS per machine
  - Software and hardware tightly coupled
  - Underutilized resources (idle time)
  - Inflexibility
- Virtualization gives you:
  - Hardware independence of operation system and applications
  - Ability to encapsulate OS and applications in to virtual machines
  - Ability to provision virtual machines to any system

# Classification of Hypervisors

There are two types of hypervisor based on architecture:

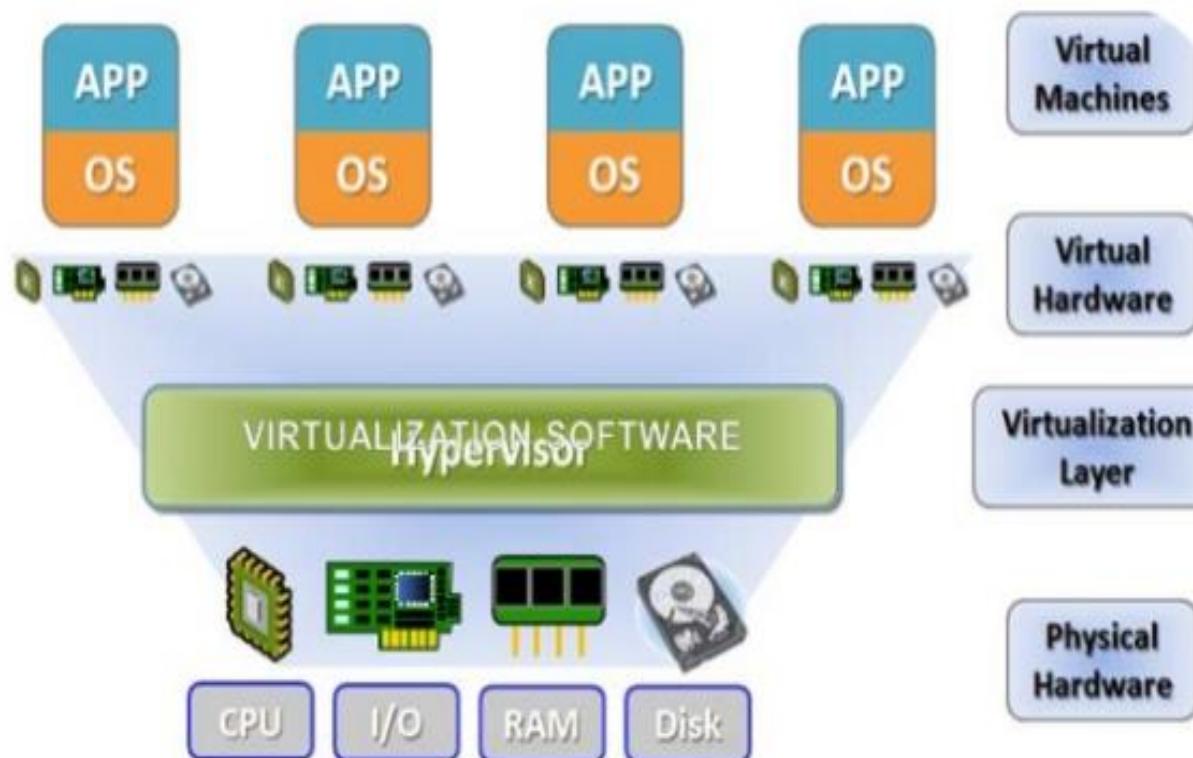
- **Type 1 Hypervisor**
  - also known as native or bare metal hypervisor
- **Type 2 Hypervisor**
  - also known as hosted hypervisor

# Technology made cloud possible



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



# How do we Get Virtualization

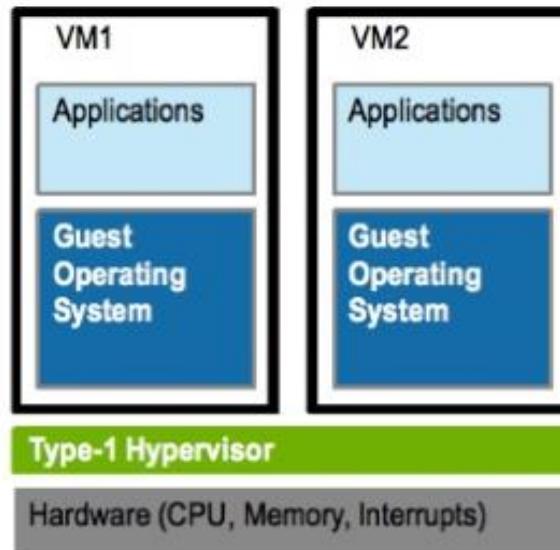
## How do we get Virtualization?

- **Hypervisors:** The approach to virtualization
- **Hypervisor** also known as the Virtual Machine Monitor
- Software that allows multiple operating systems to share a single hardware host
- Emulates hardware resources to guest operating systems
- Each operating system appears to have host's processor, memory, and other resources all to itself
- In reality, hypervisor controls the resources and allocates them as needed by each guest OS in a synchronized

# Classification of Hypervisors

## Type 1 Hypervisor

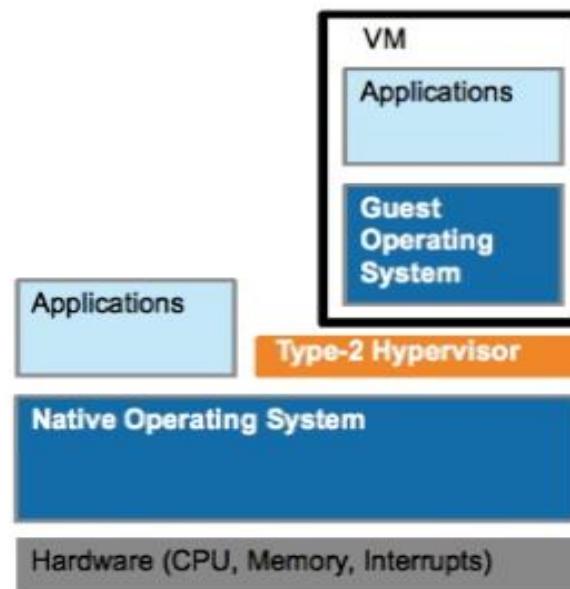
- Runs directly on the host's hardware to manage guest operating systems
- Does not require any base server operating system
- Direct access to hardware resources
- Better performance, scalability, and stability
- However, hardware support is limited



# Classification of Hypervisor

## Type 2 Hypervisor

- Hosted on the main operating system
- Basically a software installed on an OS
- Hypervisor asks OS to make hardware calls
- Better compatibility with hardware
- Increased overhead affects performance



# Classification of type I VMM

Type 1 Hypervisors can be further classification into two main ways to architect the hypervisor solutions:

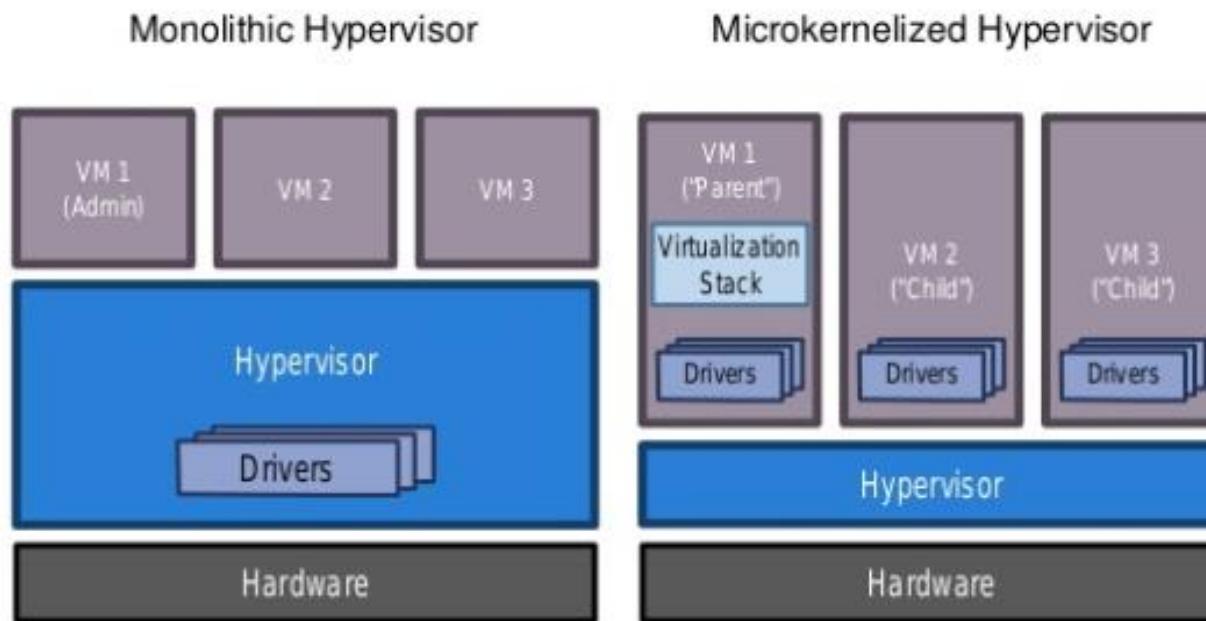
- **Monolithic**

- Hosts the hypervisor/VMM in a single layer that also includes most of the required components, such as the kernel, device drivers, and the I/O stack

- **Microkernelized**

- Uses a very thin, specialized hypervisor that only performs the core tasks of ensuring partition isolation and memory management. This layer does not include the I/O stack or device drivers.
- Virtualization stack and hardware-specific device drivers are located in a specialized partition called the parent partition.

# Classification of Type I VMM



# **Virtualization Techniques**

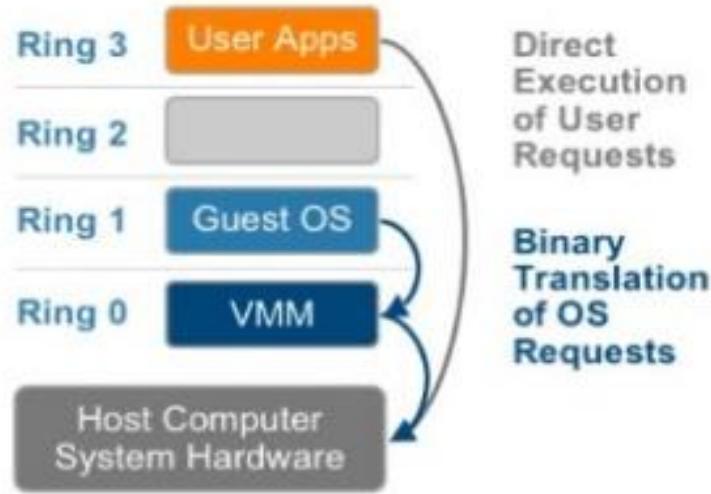
There are multiple approaches to running the guest operating system. These include:

- **Full Virtualization**
- **Paravirtualization**
  - Also known as OS assisted virtualization
- **Hardware-assisted virtualization**
  - Also known as accelerated virtualization, hardware virtual machine (HVM)

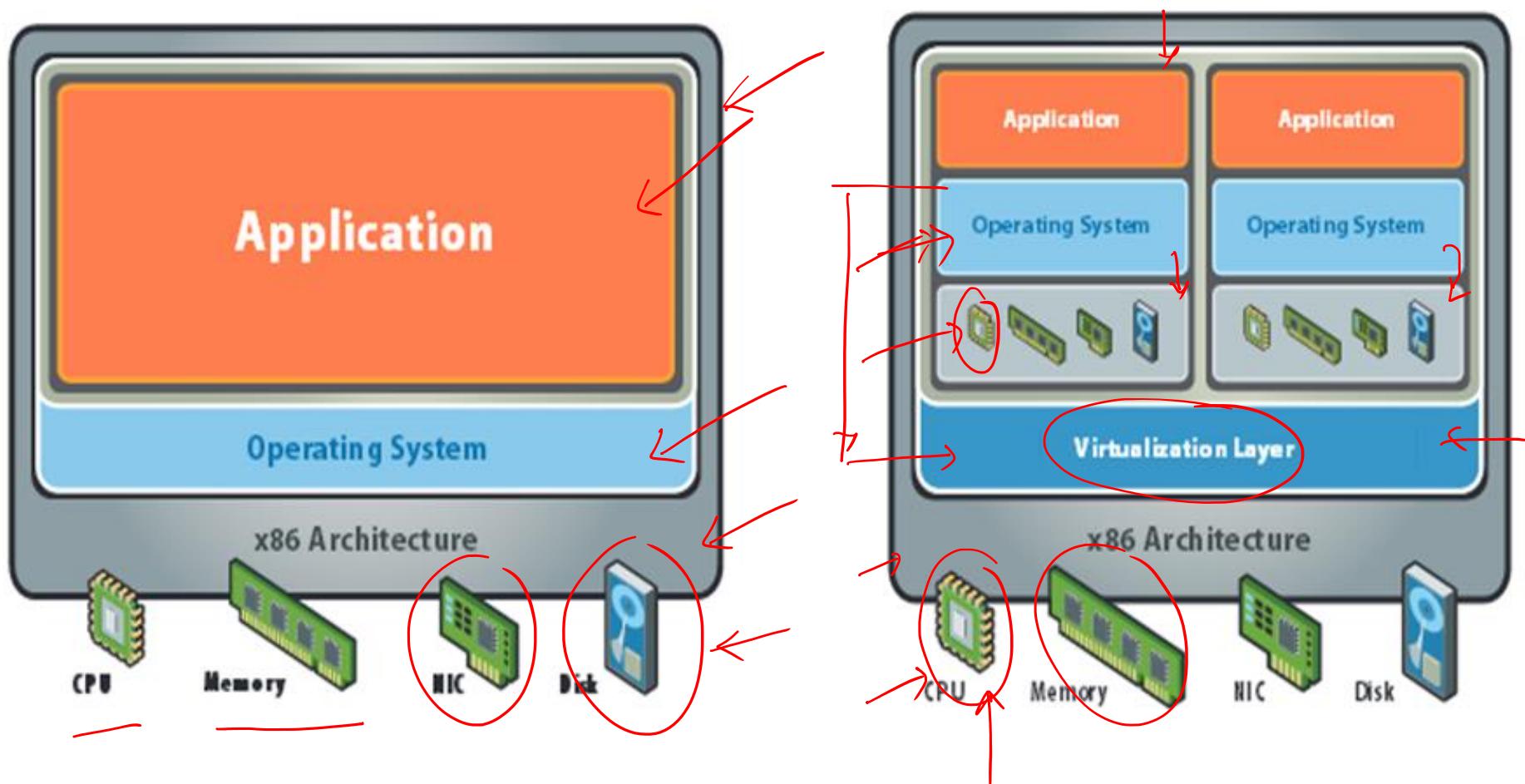
# Virtualization Techniques

## Full Virtualization

- Completely abstracted from the underlying hardware by virtualization layer
- Guest OS unaware that it is a guest
- Hypervisor translates all OS calls on-the-fly
- No hardware assistance or modification; flexibility



# What is Virtualization



# What does Virtualization do?

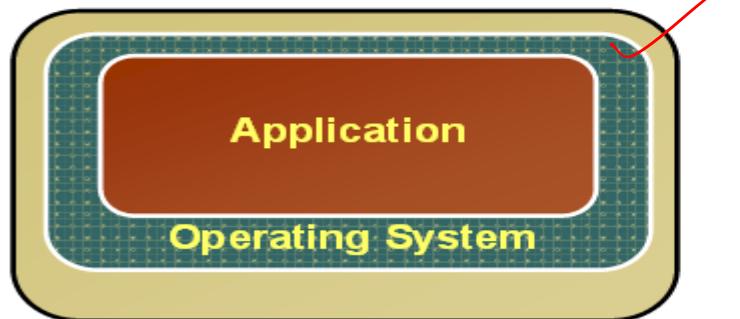
---

- 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



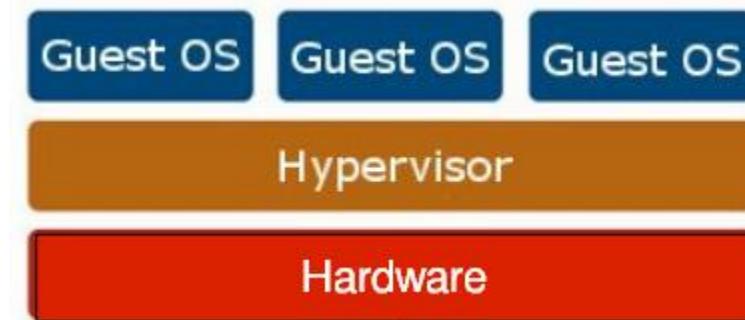
## After Virtualization

- 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



# 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 ✓

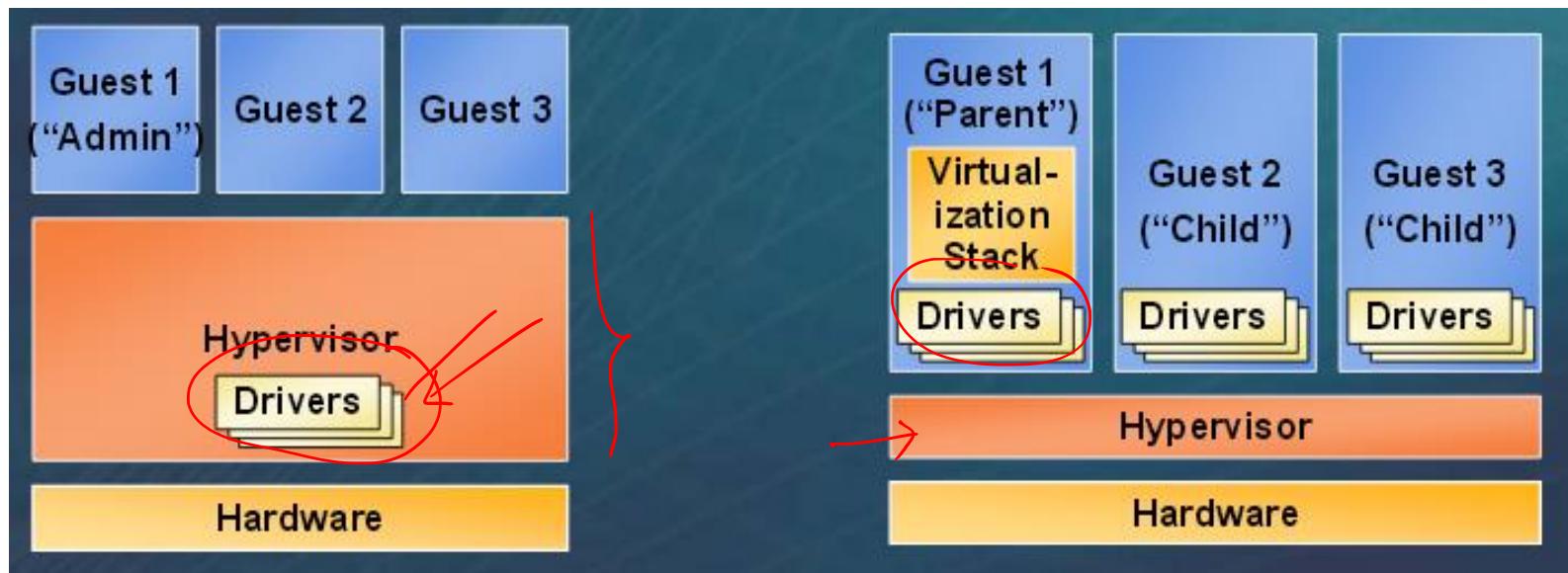
# How Hypervisor goals are achieved?

- 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

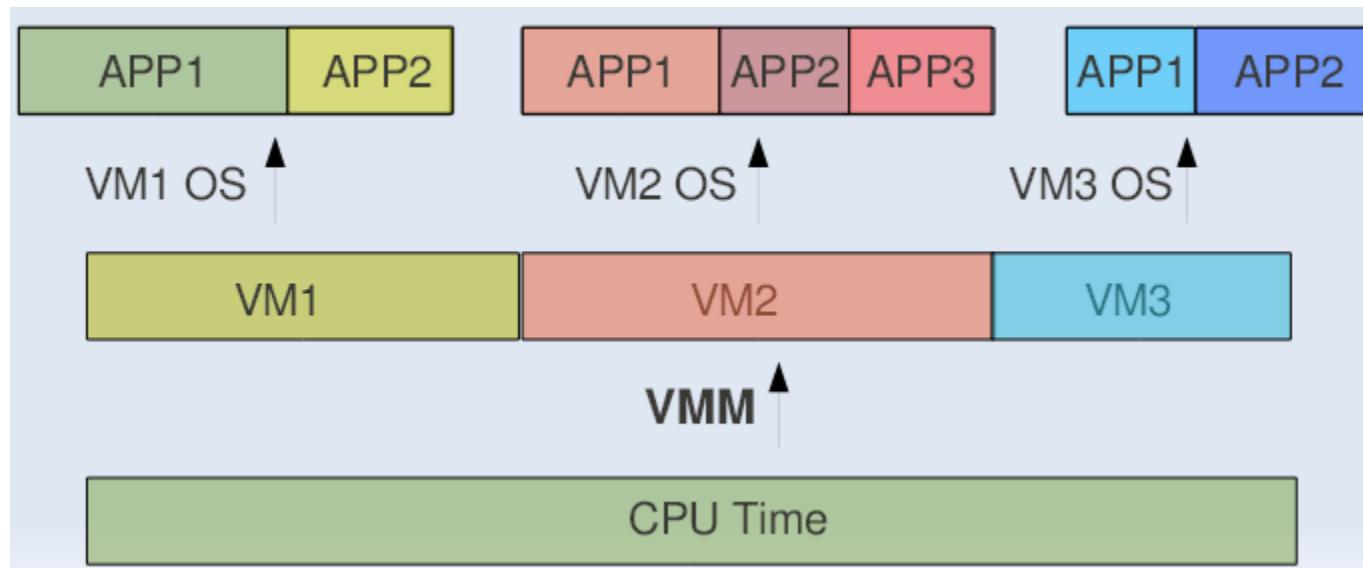
## Monolithic versus Microkernelized

- 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



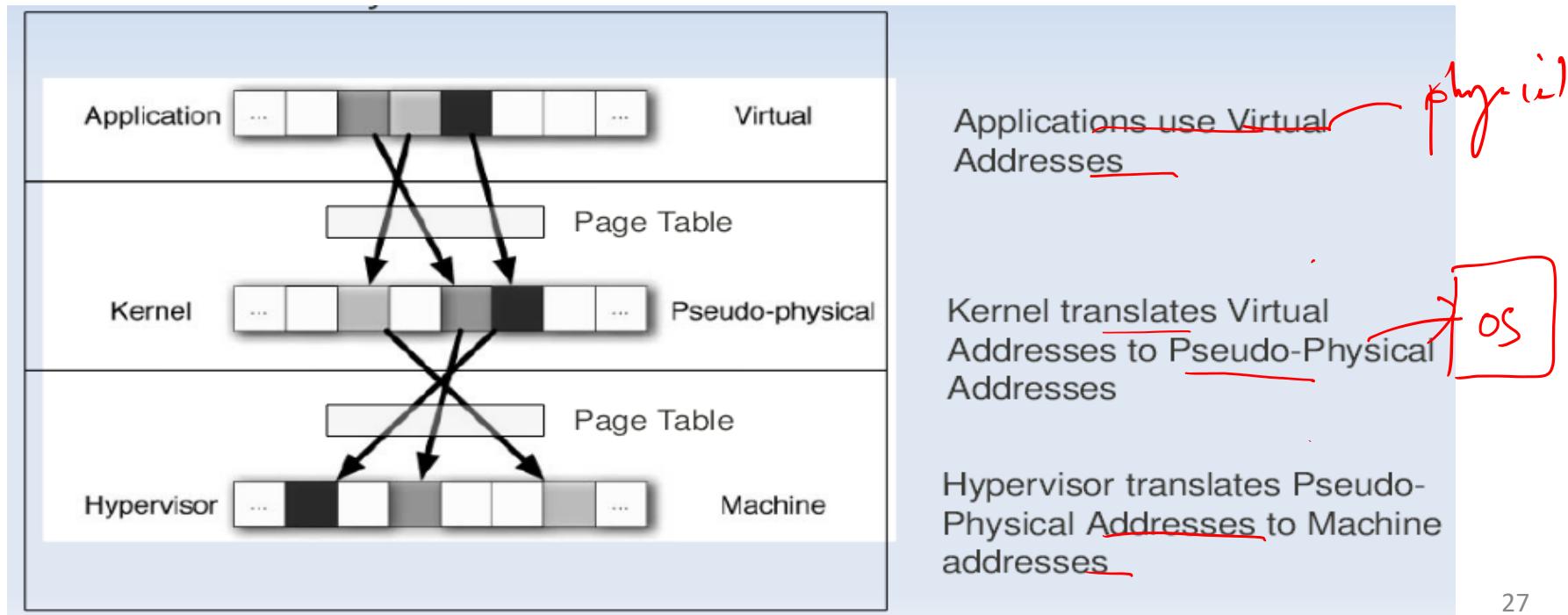
# CPU Sharing

- VMM or Hypervisor provides a virtual view of CPU to VMs.
- In multi processing, CPU is allotted 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

---

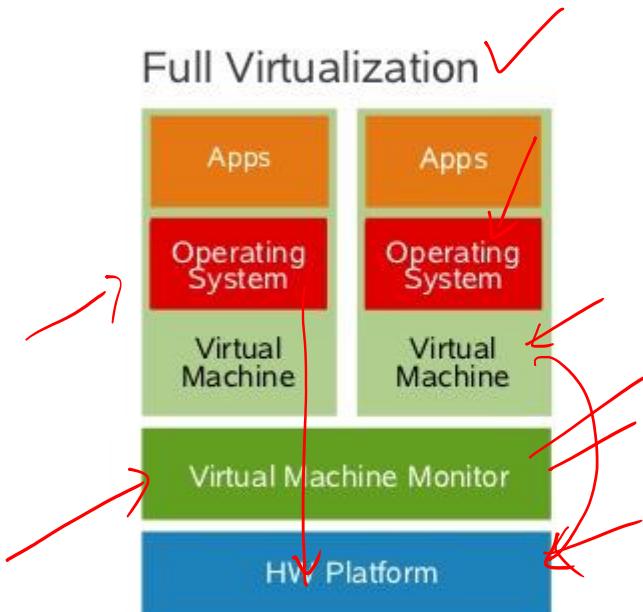
- Device needs to use Physical Memory location.
- In a virtualized environment, the kernel is running in a hypervisor-provided virtual address space
- Allowing the guest kernel to convey an arbitrary location to device for writing is a serious security hole
- Each device defines its own protocol for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

# Approaches for Virtualization

N

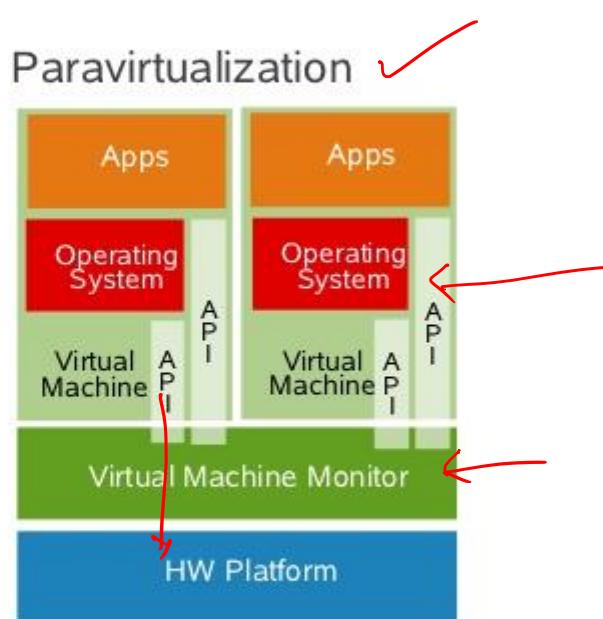
## Full & Paravirtualization Overview

Full Virtualization ✓



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

Paravirtualization ✓



Static modification of Guest OS prior to  
runtime: Privileged instruction calls are  
exchanged with API functions provided  
by the VMM

- Almost no performance degradation
- Significant scalability

# Full Virtualization

---

## ❑ Full virtualization

- In its basic form known as “full virtualization” the hypervisor provides a fully emulated machine in which an operating system can run. VMWare is a good example.
- 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.

# VM Implementation Techniques

- Binary Translation
- Paravirtualization
- Hardware Supported Virtualization

# Binary Translation

- Used in VMWare

Binary image of OS is manipulated at the runtime.

- Privileged instructions are rewritten to point to their emulated versions.
- Performance from this approach is not ideal particularly when doing anything I/O intensive.
- Caching of the locations of unsafe instructions can speed Up

# Paravirtualization

- Used in XEN
- Make OS aware of underlying Virtualization env.
- OS's code is manipulated.
- Important system calls are changed to point to the implementation provided by the VMM.

# HW Supported Virtualization

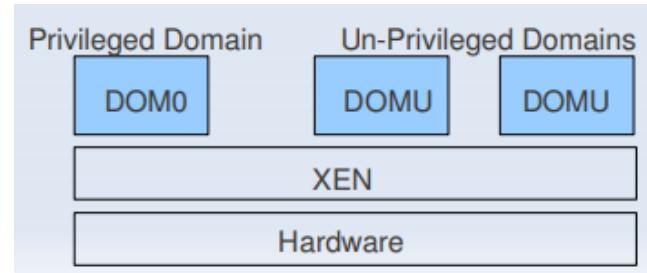
- Added new instructions which makes Virtualization considerably easier for x86.
- Intel – IVT(Intel Virtualization Technology)
- AMD – introduced AMD-V
- OS stays in its original privilege level 0.
- Attempts to access the hardware directly are caught and passed to VMM.
- In other words a new privilege ring is setup for the VMM

# XEN

- XEN Domains
- CPU Sharing
- Hyper Calls
- Memory Sharing
- IO Sharing
- XEN Split Driver Technique
- IO Ring

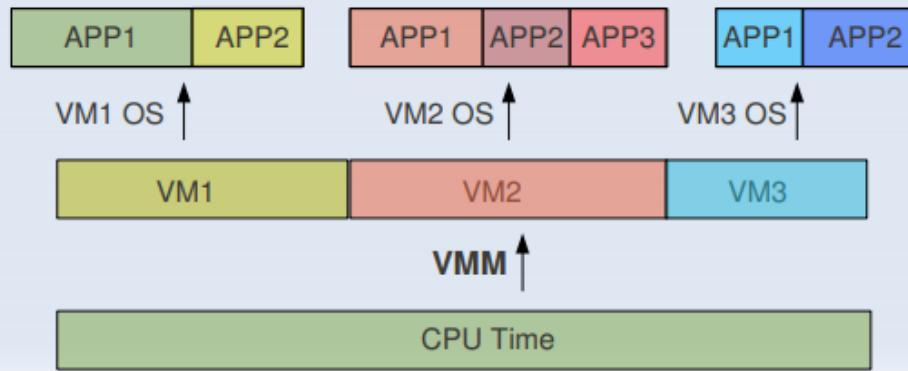
# XEN Domains

- Xen runs guests in environments known as domains which encapsulate a complete running virtual environment
- There are two types pf Domains:
- DomU -the “U” stands for unprivileged.
- Guest OSs run in this domain.
- Dom0 has elevated privileges
- Provides device drivers
- Provides tools/mechanisms to configure Virtualization environment

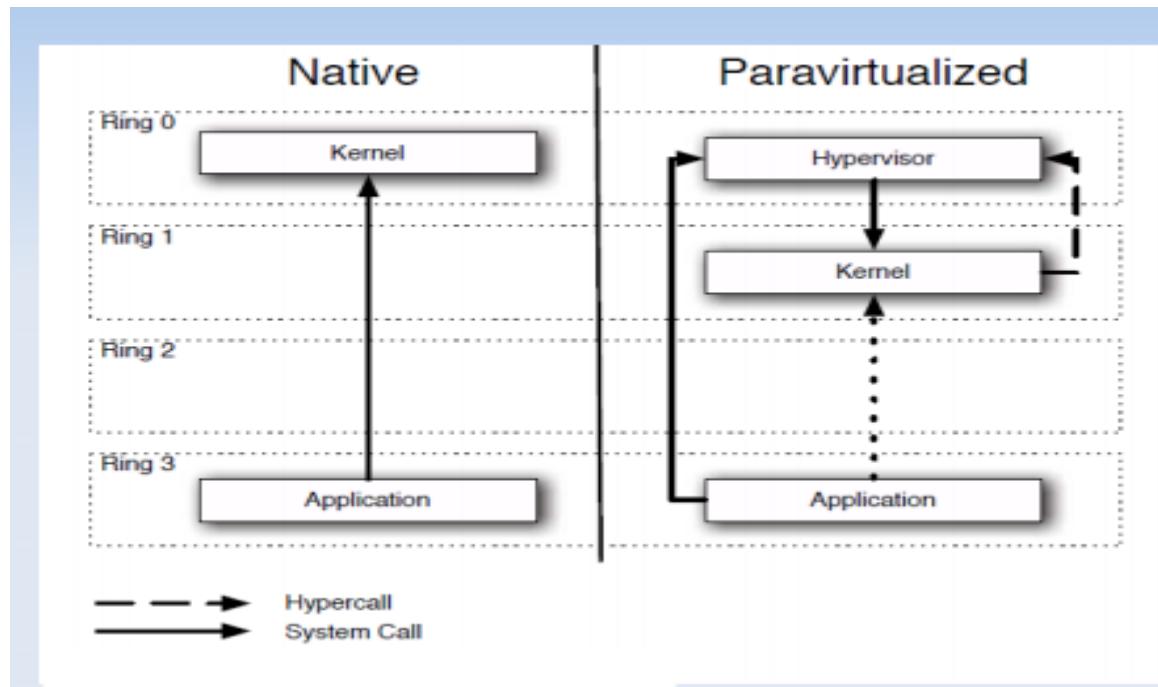


# CPU Sharing

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

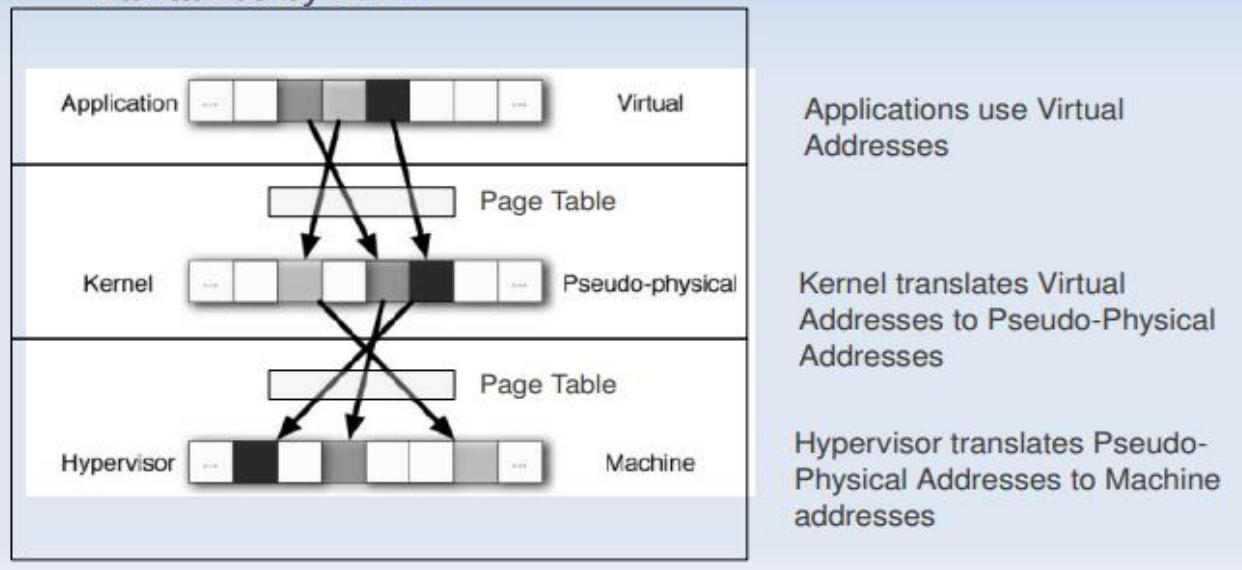


# XEN Hypervisor



# 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

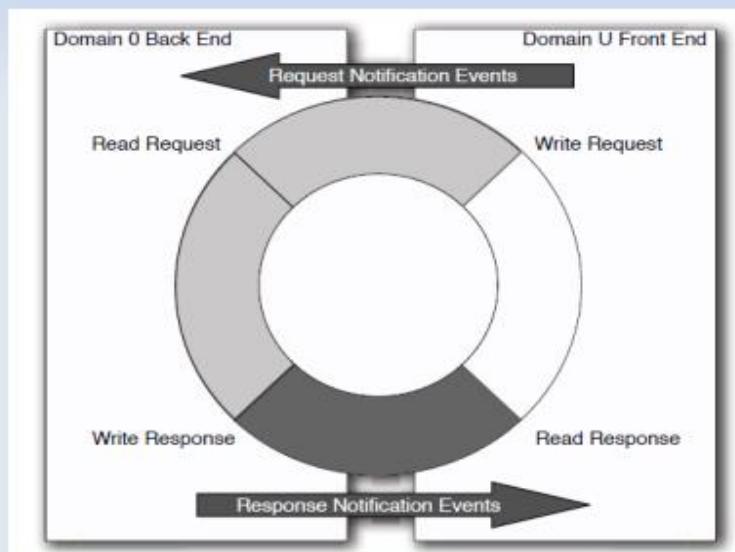


# IO Sharing

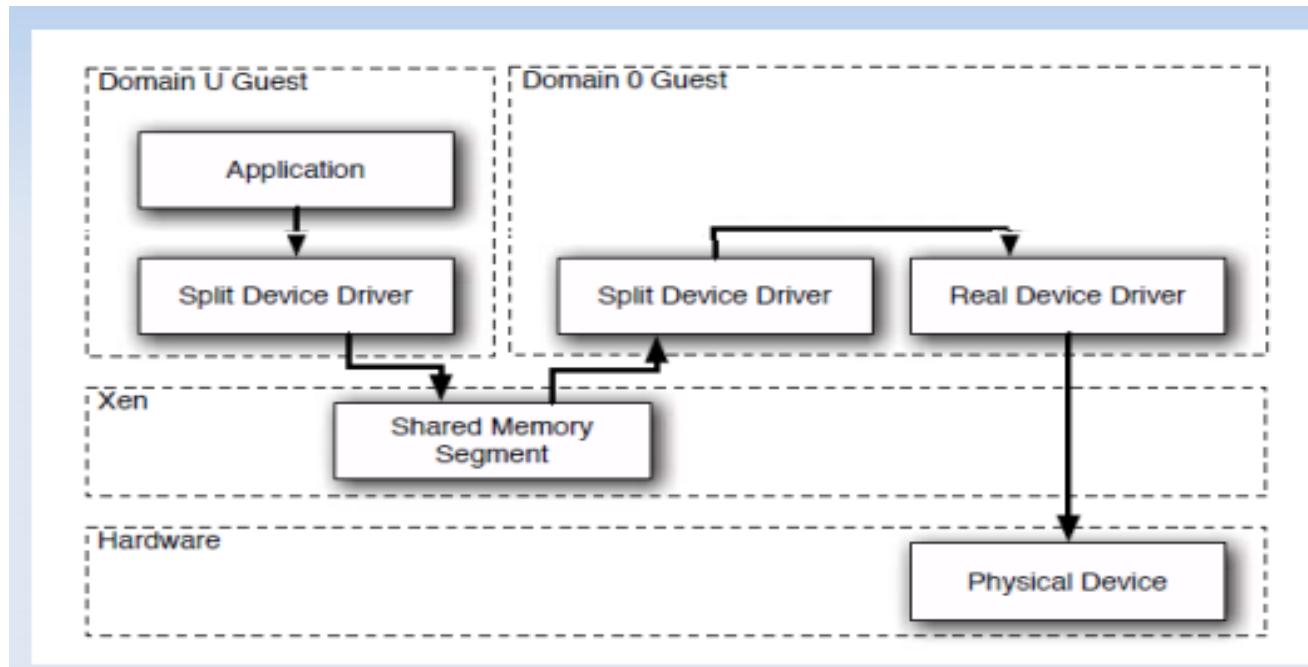
- DMA Problem
- Device needs to use Physical Memory location.
- In a virtualized environment, the kernel is running in a hypervisor provided virtual address space
- Allowing the guest kernel to convey an arbitrary location to device for writing is a serious security hole
- Detecting a DMA instruction is nontrivial. Each device defines its own protocol for talking to drivers.
- XEN Follows Split Driver Model: Dom 0 does the IO on behalf of all the other guests.
- As DOM0 is privileged the IO has no problem

# IO Ring

Shared memory is used with event based synchronization



# XEN IO Split Device Driver



# Conclusions

- Notion of Cloud is possible without Virtualization, but it will be inefficient and inflexible.
- Virtualization is an attempt to manage OS.
- There are many levels and many ways to
- implement Virtualization.

# ParaVirtualization

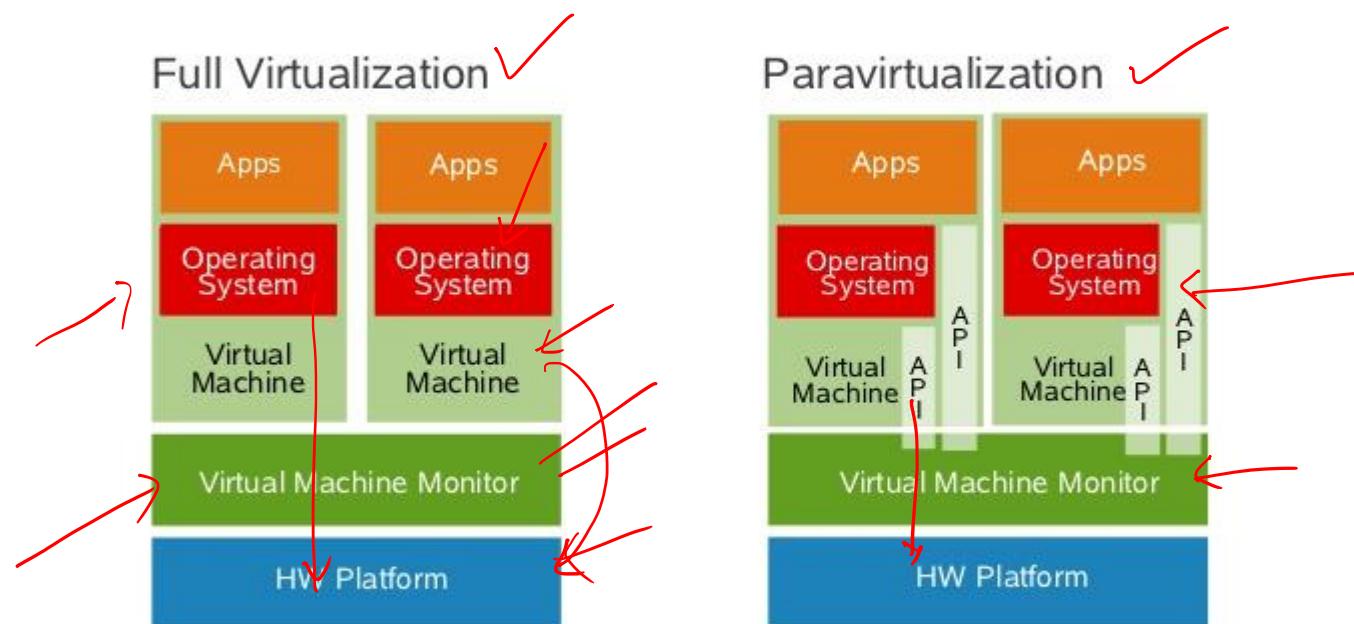
---

## □ 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

# Approaches for Virtualization

## Full & Paravirtualization Overview



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

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

# 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.

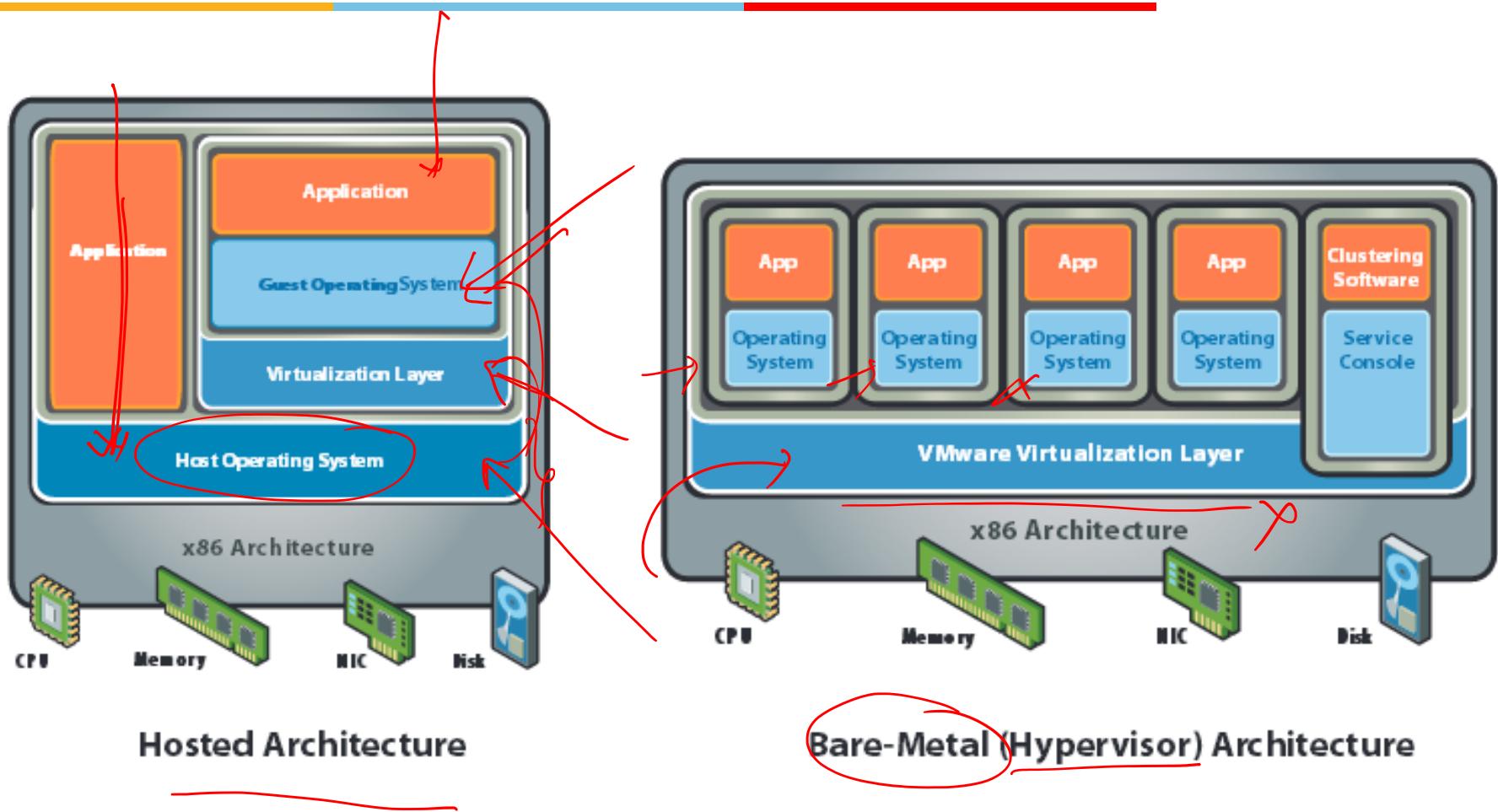
S.No.	Full Virtualization	Paravirtualization
1.	In Full virtualization, virtual machines permit the execution of the instructions with the running of unmodified OS in an entirely isolated way.	In paravirtualization, a virtual machine does not implement full isolation of OS but rather provides a different API which is utilized when OS is subjected to alteration.
2.	Full Virtualization is less secure.	While the Paravirtualization is more secure than the Full Virtualization.
3.	Full Virtualization uses binary translation and a direct approach as a technique for operations.	While Paravirtualization uses hypercalls at compile time for operations.
4.	Full Virtualization is slow than paravirtualization in operation.	Paravirtualization is faster in operation as compared to full virtualization.
5.	Full Virtualization is more portable and compatible.	Paravirtualization is less portable and compatible.
6.	Examples of full virtualization are Microsoft and Parallels systems.	Examples of paravirtualization are Microsoft Hyper-V, Citrix Xen, etc.
7.	It supports all guest operating systems without modification.	The guest operating system has to be modified and only a few operating systems support it.
8.	The guest operating system will issue hardware calls.	Using the drivers, the guest operating system will directly communicate with the hypervisor.
9.	It is less streamlined compared to para-virtualization.	It is more streamlined.
10.	It provides the best isolation.	It provides less isolation compared to full virtualization.

# 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



# 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

- Interoperability among vendor products is still evolving.
- Failure of the virtualization device, leading to loss of the mapping table.

# SKI Virtualization

---

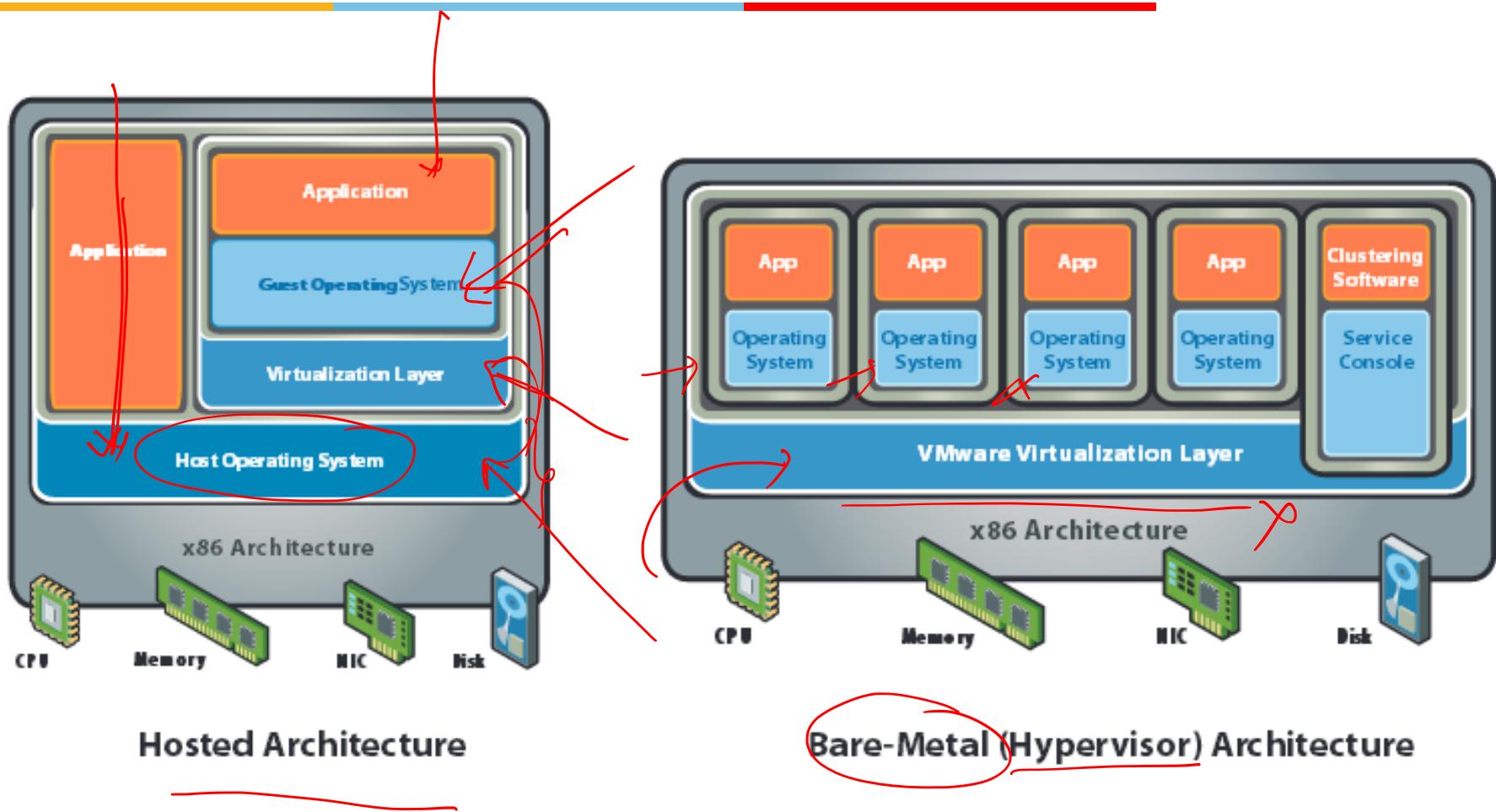
- Single Kernel Image (SKI),
  - Single Kernel Image (SKI), in which the host OS spawns additional copies of itself. This kind of virtualization can be found in Swsoft Virtuozzo and Sun Solaris, Zones. SKI can be thought of as “lightweight” virtualization.
  - While this approach avoids the performance 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



# 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

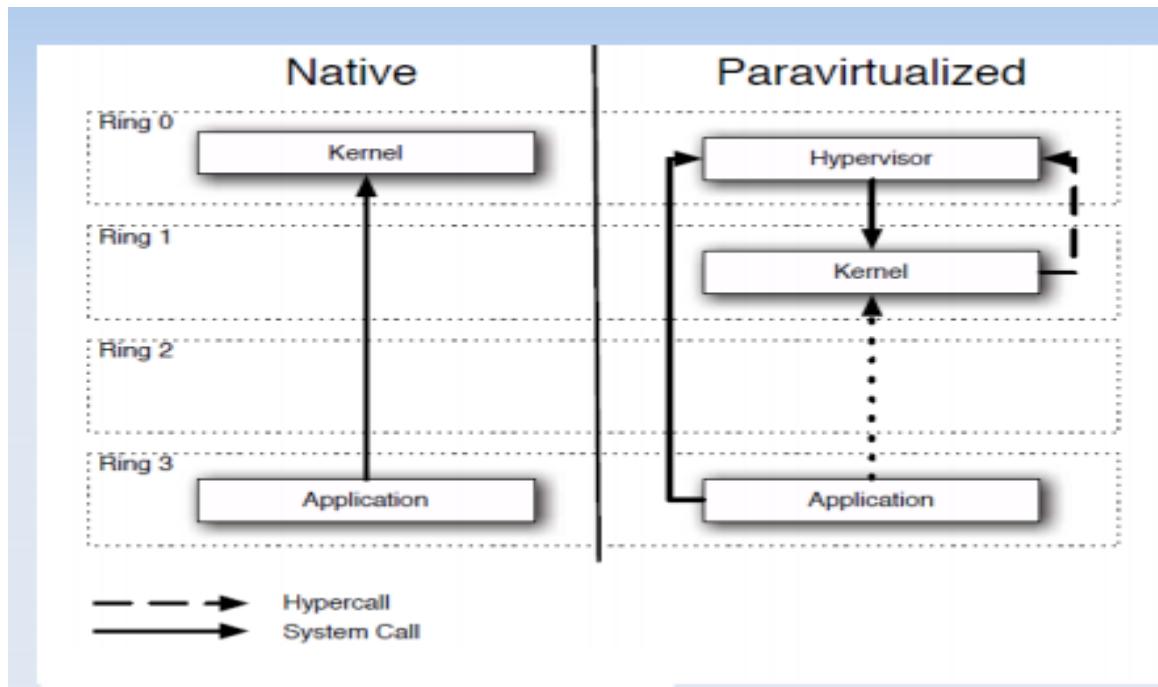
---

- 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.

# Benefits of using Virtual Machines

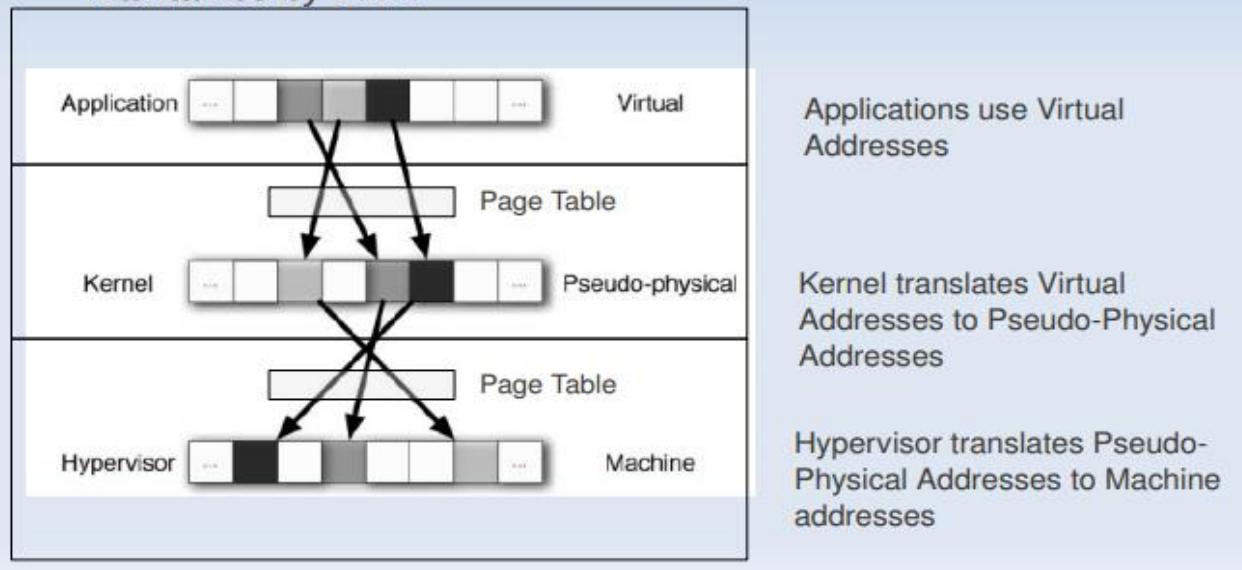
- Instant provisioning - fast scalability
- Live Migration is possible
- Load balancing and consolidation in a Data Center is possible.
- Low downtime for maintenance
- Virtual hardware supports legacy operating systems efficiently
- Security and fault isolation

# XEN Hypervisor



# 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

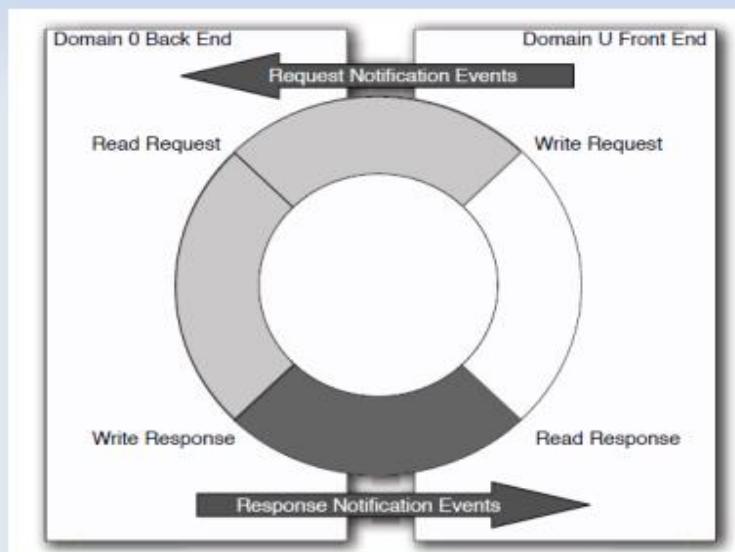


# IO Sharing

- DMA Problem
- 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
- Detecting a DMA instruction is nontrivial. Each device defines its own protocol for talking to drivers.
- XEN Follows Split Driver Model: Dom 0 does the IO on behalf of all the other guests.
- As DOM0 is privileged the IO has no problem

# IO Ring

Shared memory is used with event based synchronization



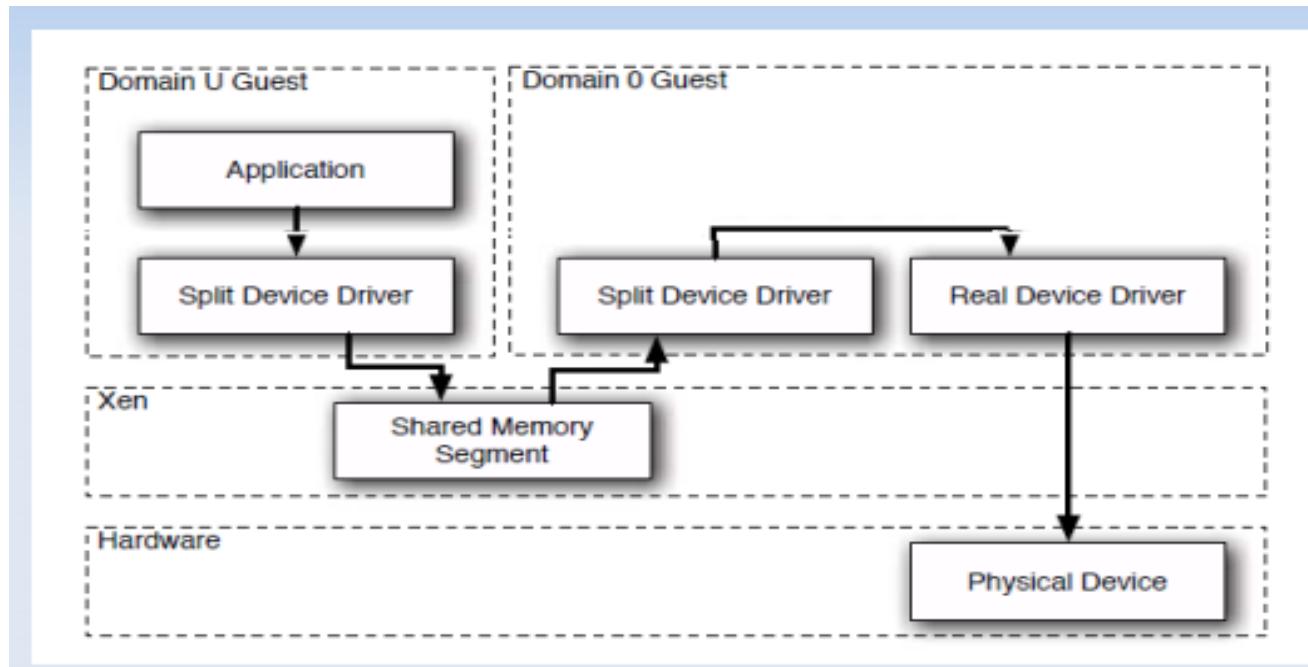
# Privilege Rings

- Memory page has a 2 bit code which is checked by CPU before executing the instruction.

If privilege level is insufficient the CPU does not execute the instruction.



# XEN IO Split Device Driver



# Conclusions

- Notion of Cloud is possible without Virtualization, but it will be inefficient and inflexible.
- Virtualization is an attempt to manage OS.
- There are many levels and many ways to
- implement Virtualization.

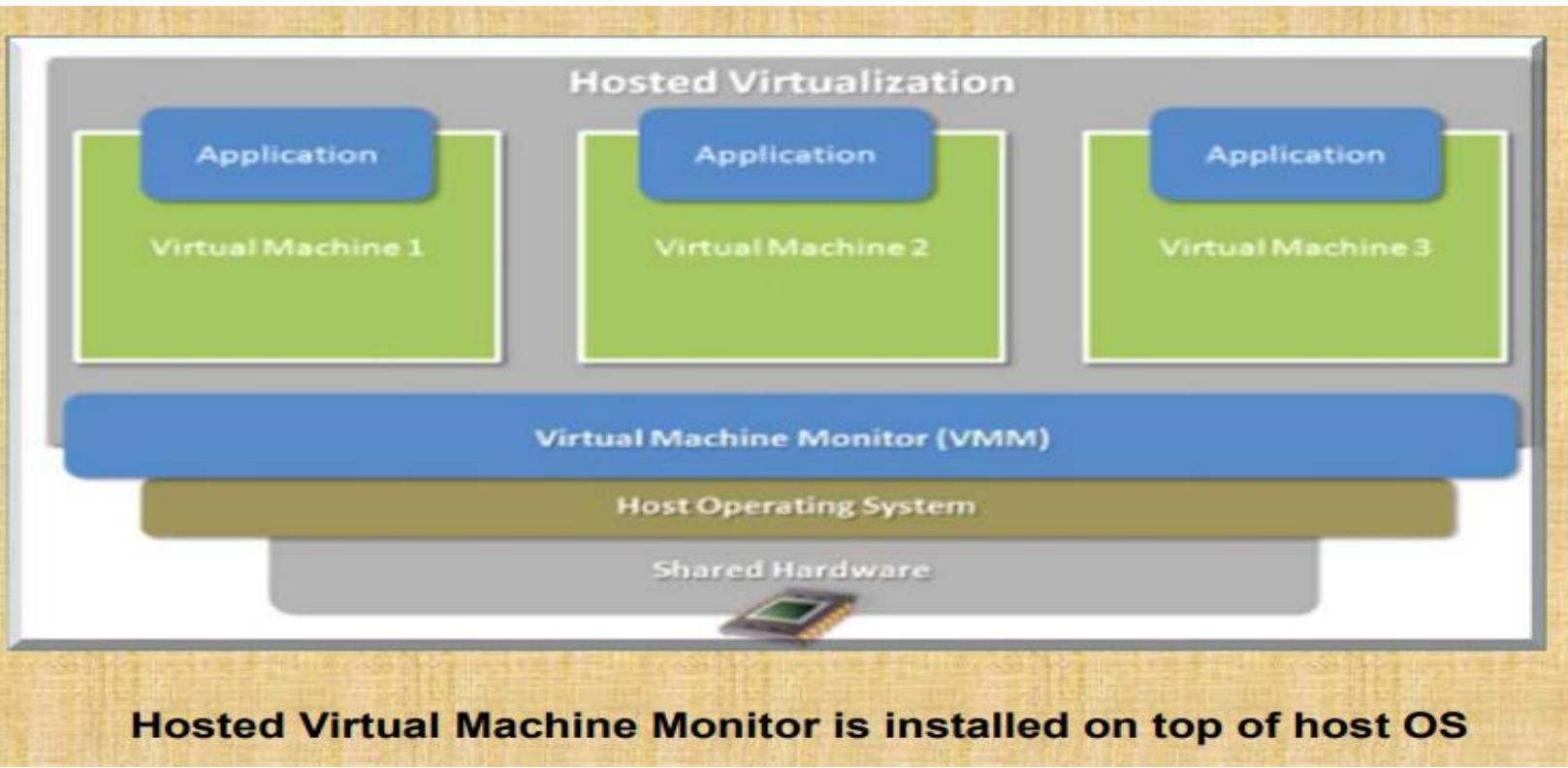
# Virtualization Architecture

- Hosted Architecture.
- Bare-Metal Architecture

# Hosted Architecture

- In this architecture, host operating system is first installed.
- A piece of software called a hypervisor or virtual machine monitor (VMM) is installed on top of the host OS.
- It allows users to run various guest operating systems within their own application windows.
- Eg. VMware Workstation, Oracle Virtual Box , Microsoft Virtual PC

# Hosted Architecture



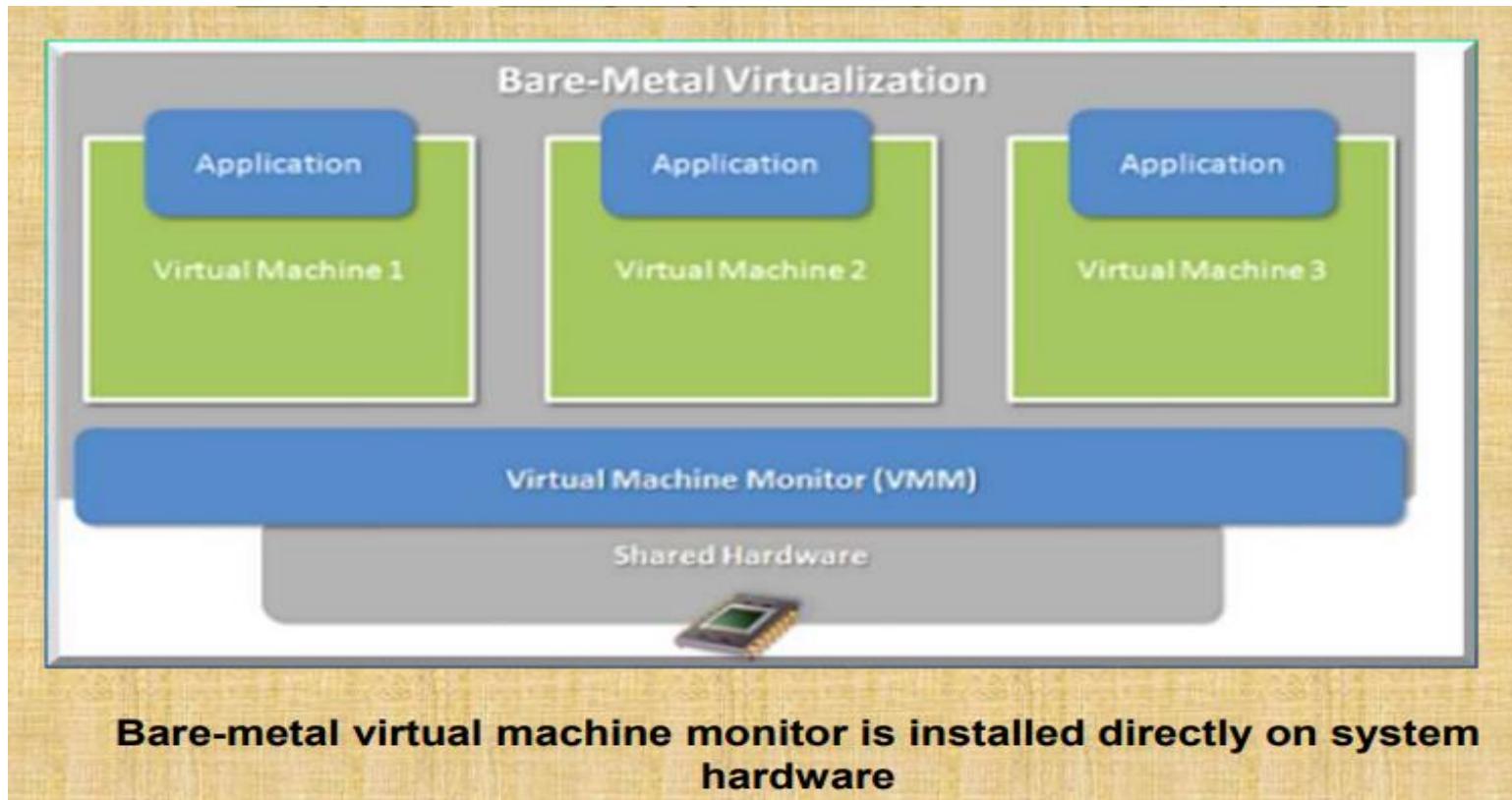
# Hosted Architecture( Pros & Cons)

- Advantage
- Ease of installation and configuration.
- Unmodified Host OS & Guest OS.
- Run on a wide variety of pc.
- Disadvantage
- performance degradation.
- lack of support for real-time operating systems.

# Bare-Metal Architecture

- In this architecture, type1 hypervisor or VMM is installed on the bare hardware.  
VMM communicates directly with system hardware rather than relying on a host operating system.
- E.g: VMWARE ESX, VMWARE ESXi, Microsoft Hyper-V

# Bare-Metal Architecture



# Bare-Metal Architecture (Pros & Cons)

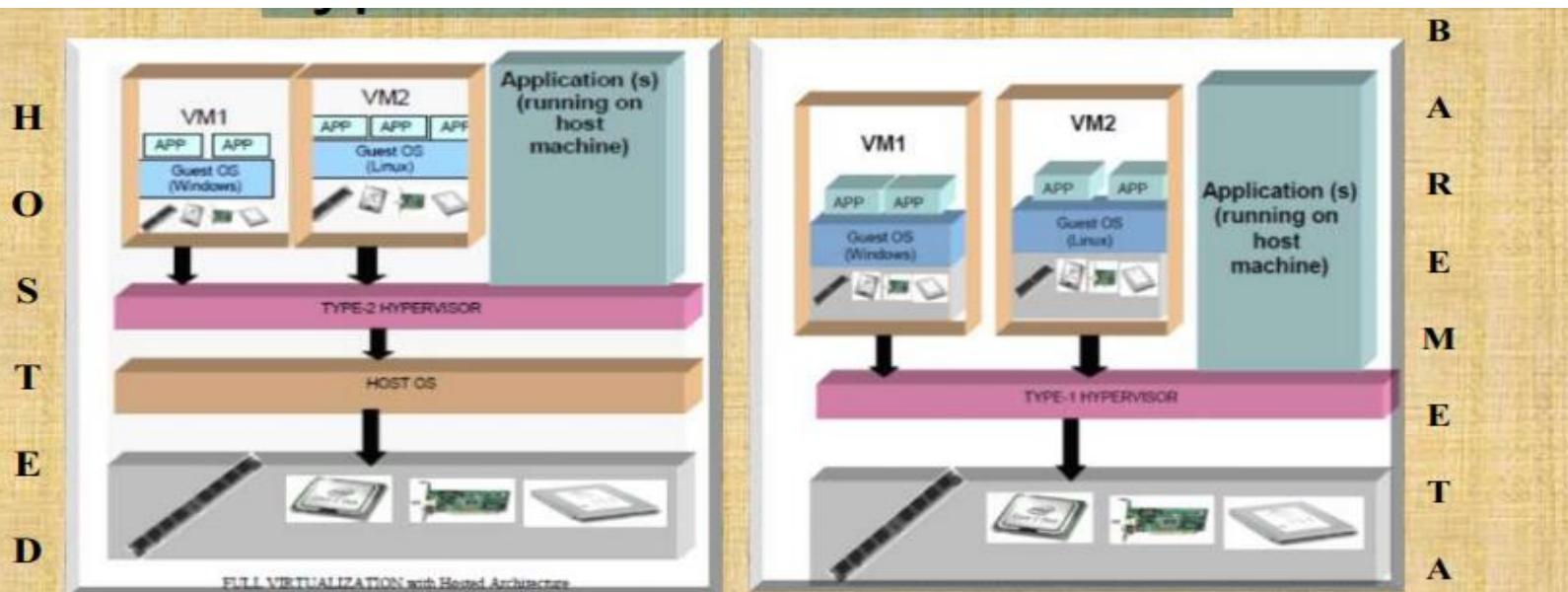
## Advantages

- Improved I/O Performance.
- Support Real Time OS.
- Disadvantage
- Difficult to install & Configure.
- Depends upon hardware platform.

## FULL VIRTUALIZATION (What is ?)

- It is a virtualization technique used to provide a virtual machine environment which is a complete simulation of the underlying hardware.
- All operating systems and applications which can run natively on the hardware can also be run in the virtual machine.
- The guest OS need not be modified.
- Guest OS do not aware the existence of VM.
- Each VM is independent of each other

# Types of Full Virtualization

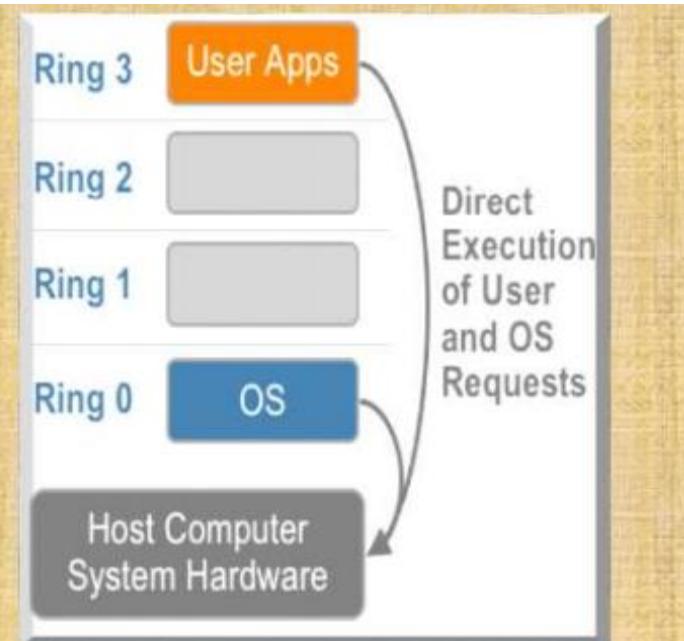


## ➤ Hypervisor or Virtual Machine Monitor (VMM)

- ✓ SW component that implements virtual machine hardware abstraction.
- ✓ Responsible for hosting and managing virtual machines & running of guest OS.

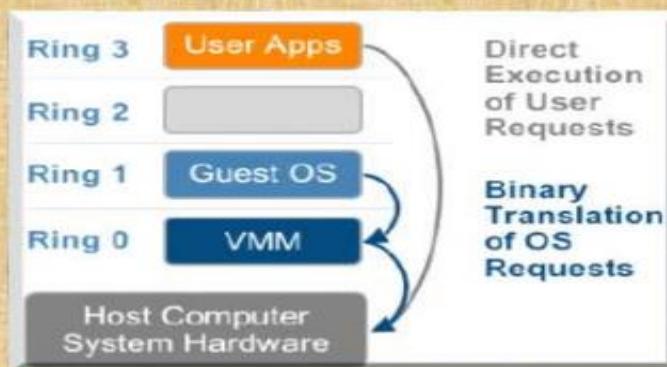
# virtualization – Challenges (X86)

- CPU provide 4 protection level(Ring 0 to Ring 3) to OS to execute code.
- OS kernel is designed to run at ring 0 to execute the code directly on the hardware and handle privileged instruction .
- User Application(s) run at ring 3 (less privileged)



**So Where Hypervisor resides?**

# Binary Translation in Full Virtualization



- VMM runs at Ring 0 & Guest OS at Ring 1 (with more privilege than application executing in user space).
- VMM executes
  - **privileged instruction** by dynamically **translating** the instruction of guest OS into a sequence of instruction appropriate to execute in real h/w.
  - It executes the **user level instruction** directly.

# Full Virtualization – Advantages & Disadvantage

## **Advantage**

- Secure
- ☐The emulation layer isolates VMs from the host OS & other application (s).
- Total VM portability
- The emulating h/w interface & guest Os forms a standard package that can be ported & run in any platform.
- Run unmodified OS
- Guest OS do not aware of being virtualized.

## **Disadvantage**

- Performance degradation in hosted full virtualization.
- Hardware dependency in bare-metal full virtualization.

# Application of Full Virtualization

- ❑ Hosted Full Virtualization is used for Desktop Virtualization.
  - ❑ Eg: Microsoft Virtual PC & Oracle VM Virtual Box.
- ❑ Bare-Metal Full Virtualization is used for Server Virtualization.
  - ❑ Microsoft Hyper-V and VMware ESX Server.
- ❑ Server Virtualization is used in Cloud Computing.

# Implementation of Full virtualization (Hosted Architecture)

## Platform

- Hardware
- Intel® Core™2 Duo CPU
- 2 GB RAM
- 160 GB HDD
- Software
- Windows 7 as Host OS.
- Windows XP & LINUX as Guest OS.
- Oracle VM Virtual Box.

# Implementation of Full virtualization (Hosted Architecture)

- Step1: Install Microsoft Virtual PC ( type2 Hypervisor).
- Step2: Create VM1 with winxp (sp2) as guest OS &
- VM2 with Linux as guest OS.
- Step3: Install your desired application on guest OS

# Conclusion

- The future of enterprise IT management will be based
- on virtual computing. Intel® VT makes it possible to
- maximize computer utilization while minimizing all
- associated overheads of management, power
- consumption, maintenance and physical space.
- Virtualization also allows the industry to run business
- with fewer machine and at reduced cost while
- providing the infrastructure to meet customer needs.

# OS and Virtualization Difference

Host OS uses the actual hardware for the working whereas the Guest OS uses the virtual hardware like number of cores and type and size of hard drive defined by the user while adding a virtual machine. ... Linux operating systems are multi-threaded operating system. The host OS would consider virtual box as a thread.

First of all there aren't any specific number of process for an OS, its called as cores or threads, technically you can define how many cores or threads you want to use on your virtual machine and it depends on the system configuration you use.

Secondly Guest OS is what you have created in the virtual machine and host is what your laptop or pc actually run. Host OS uses the actual hardware for the working whereas the Guest OS uses the virtual hardware like number of cores and type and size of hard drive defined by the user while adding a virtual machine.

Third, as I mentioned earlier Guest and Host OS works on the configurations used by you, if you user higher amount of cores/ threads in setting your virtual machine the Guest OS will get higher speed

Kernel - a program whose purpose is control and multiplexing of hardware for the benefit of other programs. ... Hypervisor - a program whose purpose is control and multiplexing of hardware for other kernels. Typically runs at an even higher privileged level than a kernel, which was invented for this purpose

# Comparison of different Server Virtualization software

- Citrix XenServer
- XenServer is an open sourced product from Citrix, based on Xen Project Hypervisor. It's a bare-metal virtualization platform with enterprise-grade features that can easily handle workloads, combined OS, and networking configurations. XenServer delivers application performance for x86 workloads in Intel and AMD environments.
- It can cater to XenApp and XenDesktop deployments, and offer customers the enhanced virtualized graphics with NVIDIA and Intel. XenServer services allow multiple computer operating systems to execute on same computer hardware

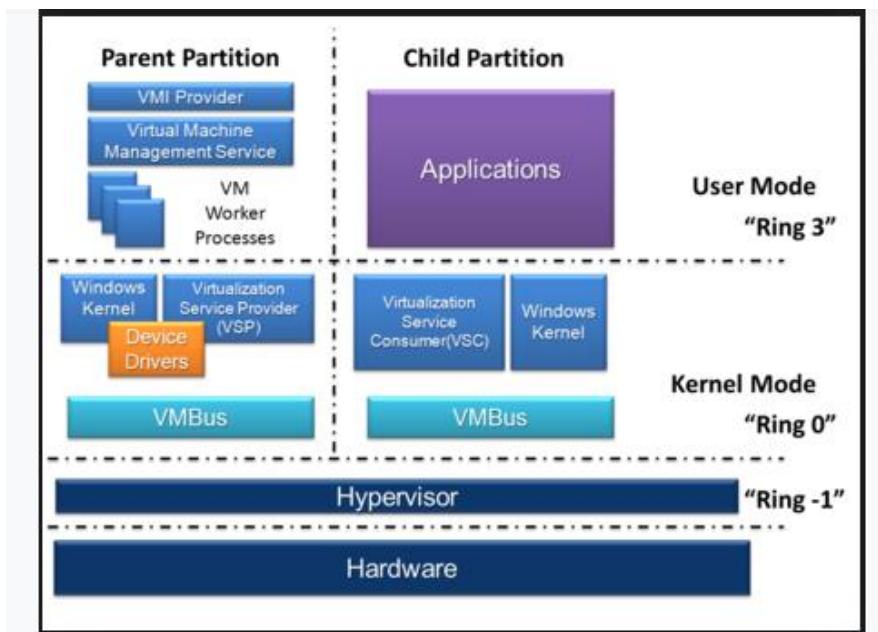
# Microsoft Hyper-V

Microsoft introduced its hypervisor in 2008, and has continued to release new versions along with the new Windows servers. Hyper -V helps one expand or establish a private cloud environment, promotes effective hardware utilization, improves business continuity and makes development and testing more efficient. We have discussed some features for Windows Server 2019 here.

# Microsoft Hyper V

## Features:

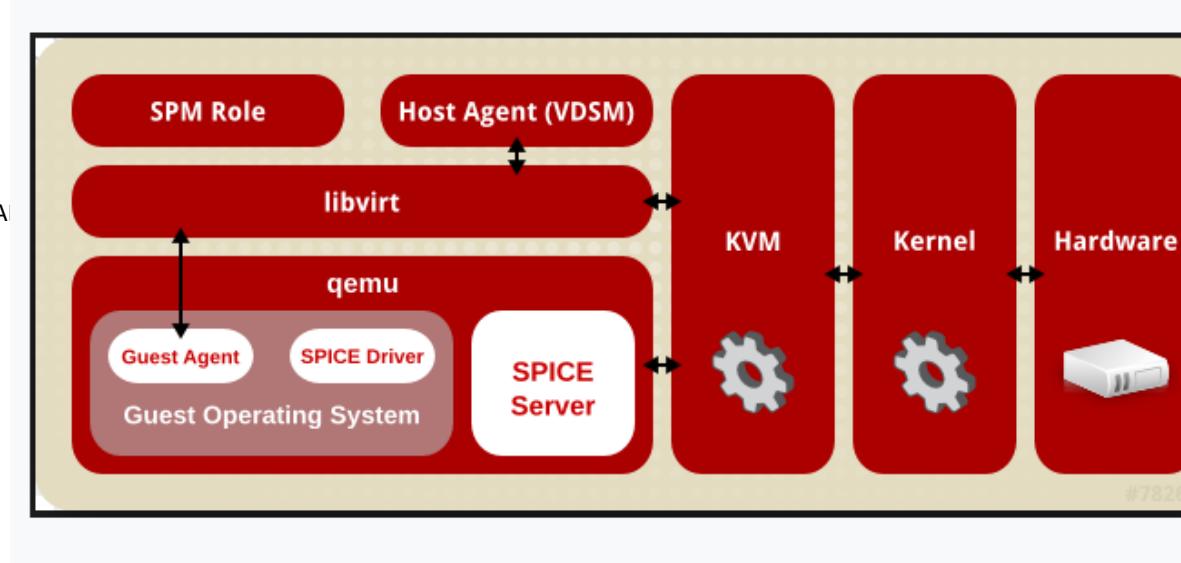
- Persistent memory support.
- Shielded VM updates.
- Simple Two-Node clusters.
- ReFS Deduplication.
- Storage Spaces Direct improvements.
- Windows Admin Center.
- Encrypted subnets.



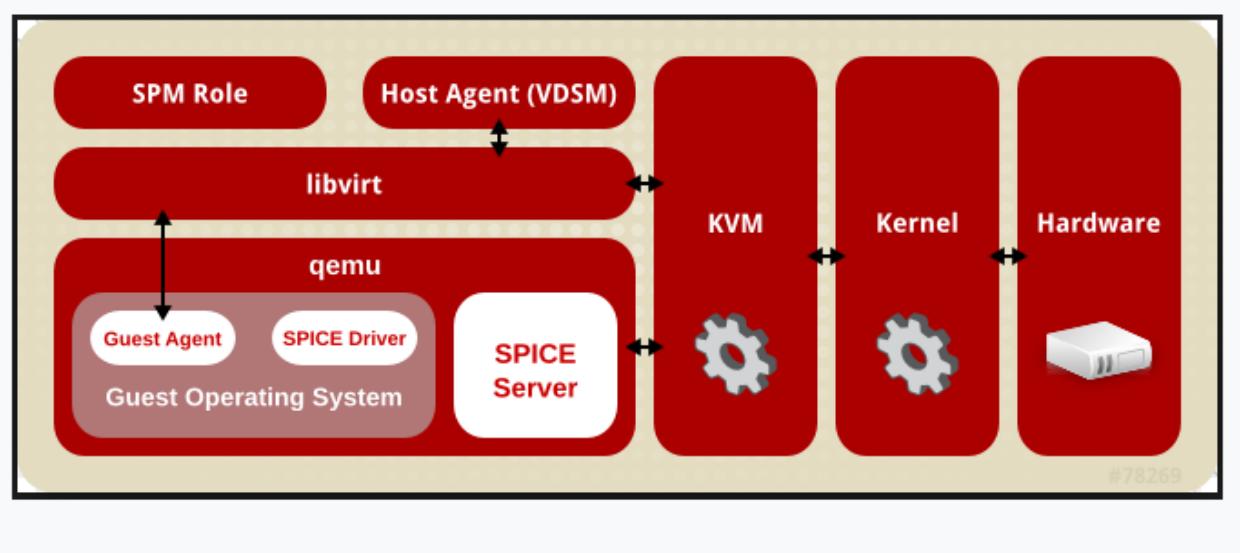
# • Red Hat KVM (Kernel-based Virtual Machine)

Red Hat's KVM is a complete virtualization infrastructure solution. Kernel-based Virtual Machine turns Linux kernel into a hypervisor. A part of Red Hat Virtualization suite, it was merged into the Linux kernel mainline in kernel version 2.6.20.

- Here is an overview of the features of KVM:
- Scalability.
- Overcommit resources.
- Disk I/O throttling.
- Hot plug of virtual resources.
- Low cost virtualization solution.
- Red Hat Enterprise Virtualization programming & API.
- Live Migration & Storage Migration.
- Assign any PCI device to virtual machines.
- Container support.
- Disaster Recovery support.
- Red Hat Satellite integration



# RedHat KVM



# VMware Vsphere

- VMware vSphere
- VMware vSphere is a set of server virtualization products that includes virtualization, management, and interface layers. It comprises following core components- infrastructure services, including VMware vCompute, vStorage and vNetwork; application services; vCenter Server – single point control across datacenter services; and clients who can access the data center via vSphere Client or via a web browser.

- **What Is a Bare Metal Server?**
- A [bare metal server](#) is a physical hosting device dedicated to a single client ([tenant](#)). Typically set up on-prem or at a third-party [data center](#) (either rented or via [colocation](#)), a bare metal server can process more data than any other hosting solution as the user has exclusive use of all computing resources, including:
  - CPU.
  - RAM.
  - Disk space.
  - [Bandwidth](#).
- Besides fully dedicated computing resources, other main reasons why companies choose a bare metal server are:
  - High levels of processing power.
  - Consistent input/output operations per second (IOPS).
  - High data privacy due to the lack of other tenants.
  - Complete control over the server's hardware and the software stack.
  - Predictable billing (typically monthly).
  - If your app is sensitive to performance and you wish to store data at a single-tenant device, the benefits of bare metal are hard to beat.

- **What Is a Virtual Machine (VM) Server?**
- A VM server is a software-based hosting setup that runs in a multi-tenant environment within a single device. Splitting a machine into individual VMs requires [server virtualization](#), a process that enables a device to host multiple systems while sharing the same physical resources (disk space, RAM, and CPU).
- Data centers create shared-resource servers using a [hypervisor](#). A hypervisor parcels the server into distinct units that rely on the same components but have separate OSs, apps, and networking. To learn more about how hypervisors and [virtualization](#) work, refer to our article on [types of hypervisors](#).
- Most companies that choose a VM server over other hosting solutions do so because of the following reasons:
  - Quick and simple deployments.
  - Flexibility to add more server resources in times of high usage and match fluctuating traffic demands.
  - Ability to control and optimize costs through a pay-as-you-go model.
  - Little to no hardware-related [server management](#).
  - Quick and reliable [snapshots and backups](#).
- VM servers are ideal for dynamic workloads and non-mission-critical apps that prioritize flexibility over consistently high performance

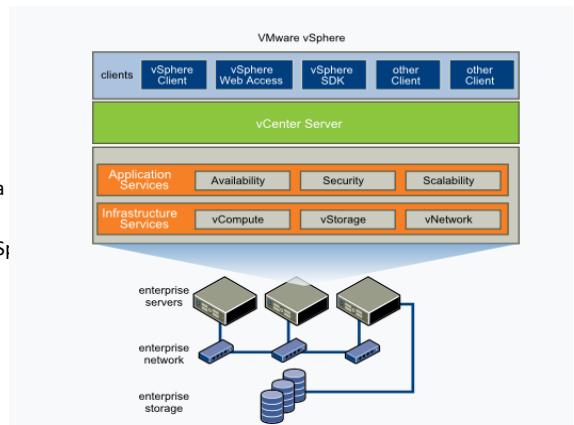
## WHY HOST ON A BARE METAL SERVER?

- No resource sharing with other server tenants
- Top, consistent performance
- High levels of security and data privacy
- The ability to set up hardware from scratch
- Complete freedom over the software stack

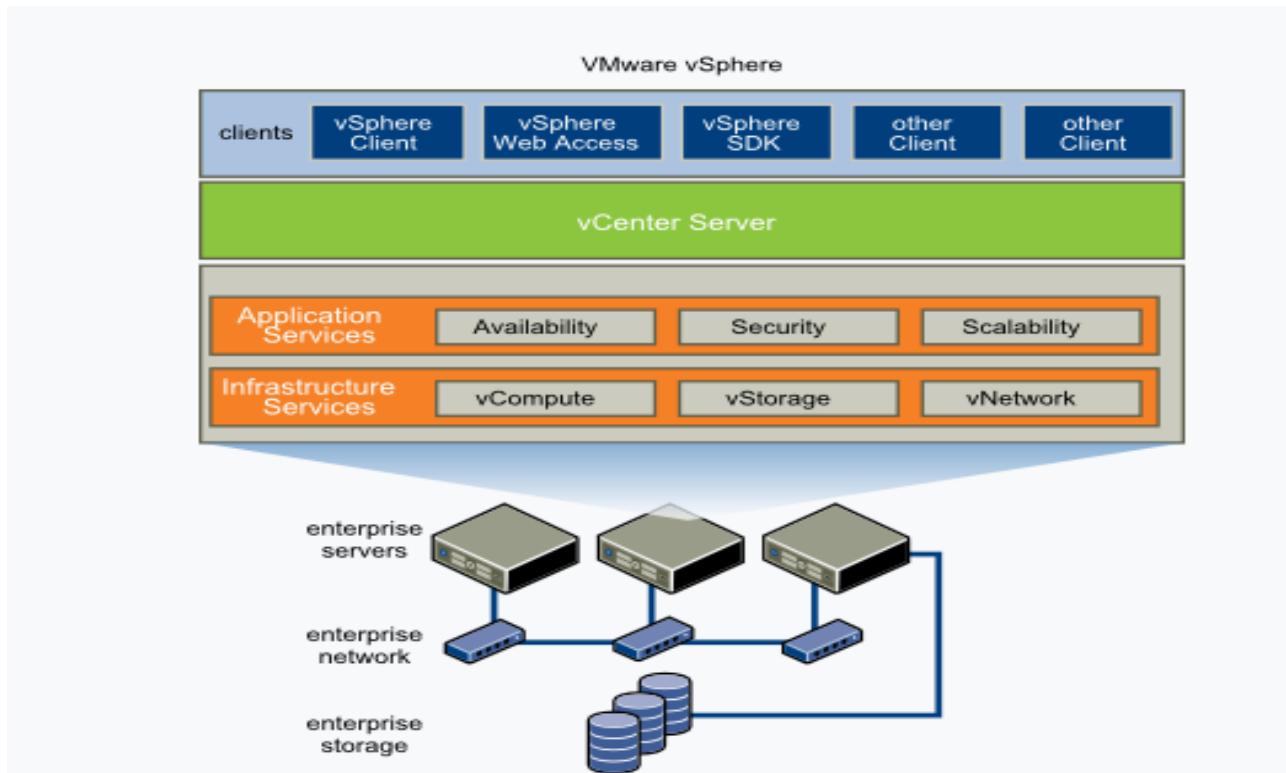




- Features and components:
- It abstracts memory, processors, storage and other resources into multiple VMs.
- vCenter Server: Centralized management tool to configure, provision and manage virtual IT environments. Provides data hosts.
- vSphere Client: vSphere 6.7 has the final version of Flash-based vSphere Web Client. Newer workflows in the updated vSphere library, vSAN, Storage policies, Host profiles, VMware vSphere Distributed Switch™ topology diagram and Licensing.
- vSphere SDKs: Provides interfaces for third-party solutions to access vSphere.
- VM File System: A cluster file system for VMs.
- Virtual SMP: Enables a single VM to use multiple physical processors at a time.
- vMotion: Enables live migration with transaction integrity.
- Storage vMotion: Enables VM file migration from one place to other without service interruption.
- High Availability: If one server fails, VM is shifted to another server with spare capacity to enable business continuity.
- Distributed Resource Scheduler (DRS): Assigns and balances compute automatically across hardware resources available for VMs.
- Fault Tolerance: Generates copy of primary VM to ensure its continuous availability.
- Distributed Switch (VDS): Spans multiple ESXi hosts and enables considerable reduction of network maintenance activities and increases network capacity.
- Network & Storage I/O Control.
- Hot add CPU and RAM resources



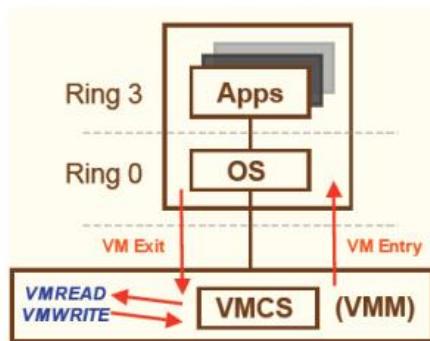
# VMware vSphere



# Hypervisor Comparison 2019: KVM vs Hyper-V vs XenServer vs vSphere

Feature	Windows Hyper-V 2019	vSphere 6.7	XenServer 7.6	KVM
RAM/Host	24TB	12 TB	5TB	12TB
RAM/VM	12 TB for generation 2;	6 TB	1.5TB	6 TB
1 TB for generation 1				
CPUs/VM	240 for generation 2;	128	32	240
64 for generation 1;				
VM Disk	64 TB for VHDX format;	62TB	2TB	10TB
2040 GB for VHD format				
VM Live Migration	Yes	Yes	Yes	Yes
VM Replication supports	Yes	Yes	Yes	Yes
Overcommit resources	No	Yes	No	Yes
Disk I/O Throttling	Yes	Yes	Yes	Yes
Hot plug of virtual resources	Yes	Yes	Yes	Yes

# VM Control Structure



## The VM Control Structure (VMCS)

VM-execution controls	Determines what operations cause VM exits	CR0, CR3, CR4, Exceptions, IO Ports, Interrupts, Pin Events, etc.
Guest-state area	Saved on VM exits Reloaded on VM entry	EIP, ESP, EFLAGS, IDTR, Segment Regs, Exit info, etc.
Host-state area	Loaded on VM exits	CR3, EIP set to monitor entry point, EFLAGS hardcoded, etc.
VM-exit controls	Determines which state to save, load, how to transition	Example: MSR save -load list
VM-entry controls	Determines which state to load, how to transition	Including injecting events (interrupts, exceptions) on entry

# Virtualization Benefits

## **REDUCE ENERGY COSTS AND GO GREEN WITH VMWARE VIRTUALIZATION**

Reduce the energy demands of your datacenter by dynamic management of computer capacity across a pool of servers.

VMware infrastructure delivers the resources your infrastructure needs and enables you to:

- Reduce energy costs by 80%.
- Power down servers without affecting applications or users.
- Green your datacenter while decreasing costs and improving service levels.

## **What Is a Bare Metal Server?**

A [bare metal server](#) is a physical hosting device dedicated to a single client (**tenant**). Typically set up on-prem or at a third-party [data center](#) (either rented or via [colocation](#)), a bare metal server can process more data than any other hosting solution as the user has exclusive use of all computing resources, including:

- CPU.
- RAM.
- Disk space.
- [Bandwidth](#).

Besides fully dedicated computing resources, other main reasons why companies choose a bare metal server are:

- High levels of processing power.
- Consistent input/output operations per second (IOPS).
- High data privacy due to the lack of other tenants.
- Complete control over the server's hardware and the software stack.
- Predictable billing (typically monthly).

## WHY HOST ON A BARE METAL SERVER?

- No resource sharing with other server tenants
- Top, consistent performance
- High levels of security and data privacy
- The ability to set up hardware from scratch
- Complete freedom over the software stack



## **What Is a Virtual Machine (VM) Server?**

A VM server is a software-based hosting setup that runs in a multi-tenant environment within a single device. Splitting a machine into individual VMs requires [server virtualization](#), a process that enables a device to host multiple systems while sharing the same physical resources (disk space, RAM, and CPU).

Data centers create shared-resource servers using a **hypervisor**. A hypervisor parcels the server into distinct units that rely on the same components but have separate OSs, apps, and networking. To learn more about how hypervisors and [virtualization](#) work, refer to our article on [types of hypervisors](#). Most companies that choose a VM server over other hosting solutions do so because of the following reasons:

- Quick and simple deployments.
- Flexibility to add more server resources in times of high usage and match fluctuating traffic demands.
- Ability to control and optimize costs through a pay-as-you-go model.
- Little to no hardware-related [server management](#).
- Quick and reliable [snapshots and backups](#).

VM servers are ideal for dynamic workloads and non-mission-critical apps that prioritize flexibility over consistently high performance.

# WHY HOST ON A VIRTUAL SERVER?



- Near-instant server and component deployment
- Ideal for dynamic, unpredictable workloads
- On-demand scalability
- Pay only for resources you use
- Quick and reliable disaster recovery

## Bare Metal Vs VM Servers (Comparison Table)

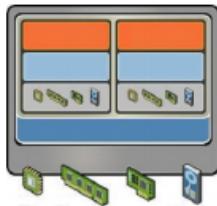
The table below offers a high-level overview of the main differences between bare metal and VM servers:

POINT OF COMPARISON	BARE METAL SERVER	VM SERVER
<b>Main selling points</b>	Consistent performance and complete data privacy	Near-instant scalability and cost optimization options
<b>Hardware dedication</b>	All server resources (CPU, RAM, memory, bandwidth) belong to a single user	Tenants host on the same device and share server resources
<b>Performance capabilities</b>	Consistently high performance	Less consistent performance due to multiple tenants
<b>Customization options</b>	The tenant has complete freedom when setting up both hardware and software	Fewer software customization options due to the shared nature of the server
<b>Deployment time</b>	Takes time to set up a new server (hours for a rented device, days for an on-prem server)	New deployments are a matter of a few minutes
<b>Hardware maintenance</b>	Complex without managed services	No hardware maintenance
<b>Scalability</b>	Scaling up or down requires months for on-prem servers, hours for rented devices	Near-instant, on-demand scalability (both up and down)
<b>Capacity optimization</b>	Limited capacity optimization	Advanced capacity optimization enabled by <a href="#">load balancing</a>
<b>Security</b>	Customization options and single tenancy ensure a secure IT platform	Other tenants can cause security and privacy concerns
<b>System recovery</b>	Recovering from a mid-size disaster can take hours or even days	Recovering from a disaster happens in minutes
<b>Server portability</b>	Moving a physical server is a complex and lengthy task	You can easily move a VM across virtual environments or from one physical server to another
<b>Typical billing methods (for a rented server)</b>	A predictable (typically monthly) bill	Charges based on how much resources you use
<b>On-prem expenses</b>	High upfront costs for hardware and space (but no need for licensing purchase)	Smaller hardware costs but pricey VM software licenses

# VMware Server Virtualization

## Features

### KEY FEATURES OF THE VMWARE SERVER VIRTUALIZATION



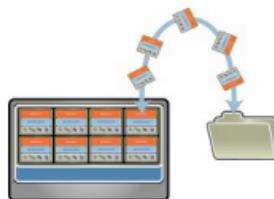
#### Partitioning

- Different OS can run on one physical machine
- System resources can be divided between virtual machines



#### Isolation

- Fault and security isolation on a hardware level
- Extended resource control for constant performance

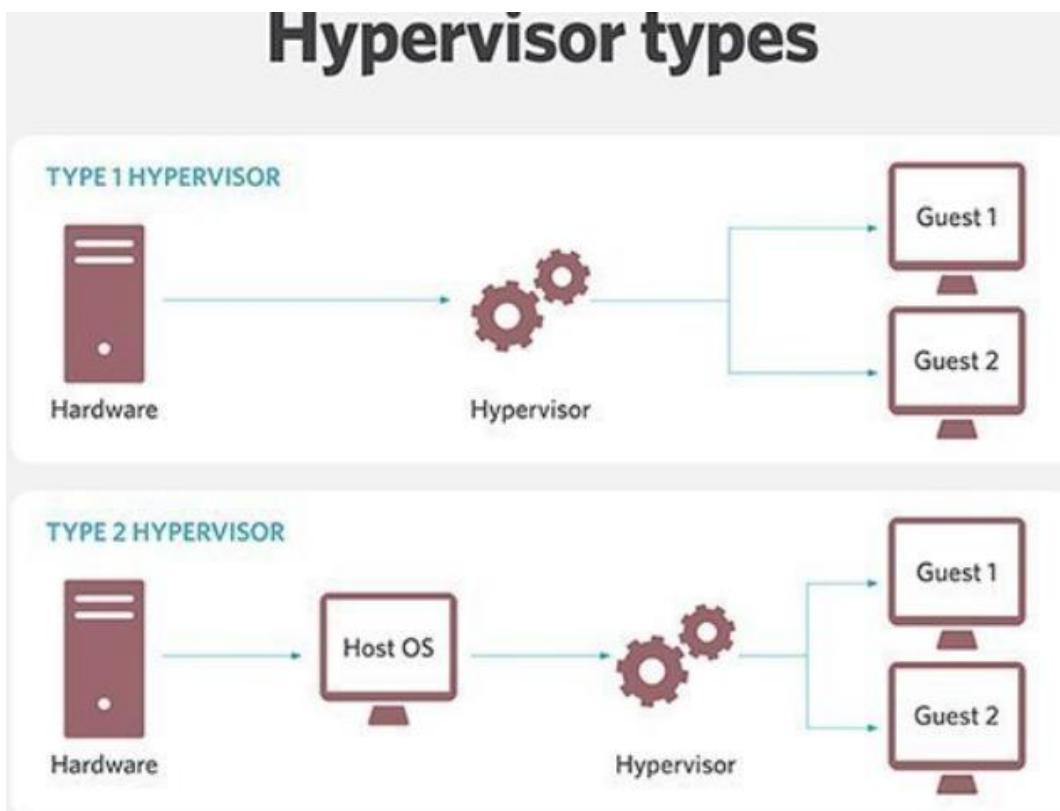


#### Encapsulation

- Complete status of a virtual machine can be stored in a file
- Move and copy of a virtual machine is as easy as it is with files



# Hypervisor types





**BITS** Pilani  
Pilani Campus

# BITS Pilani presentation

Mridul Moitra  
Cloud Computing



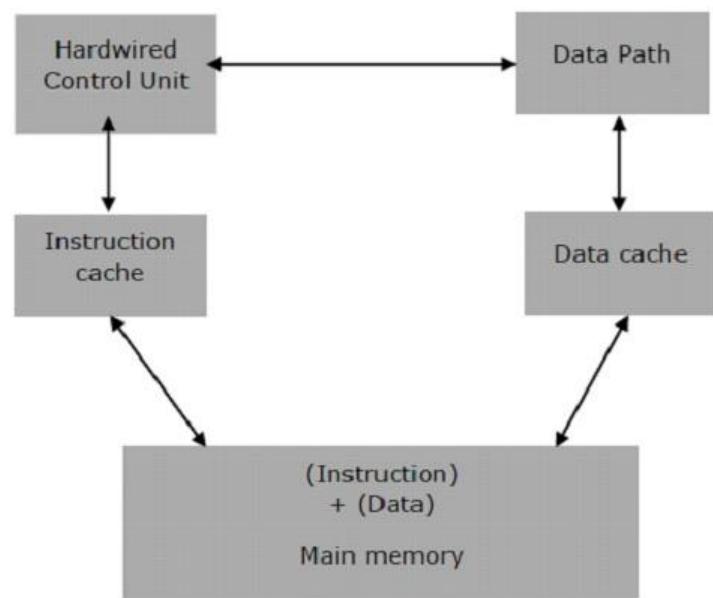


<CSI ZG527 / SS ZG527 / SE ZG527  
Cloud Computing  
Lecture No. 3

**BITS** Pilani

# Difference between RISC and CISC -

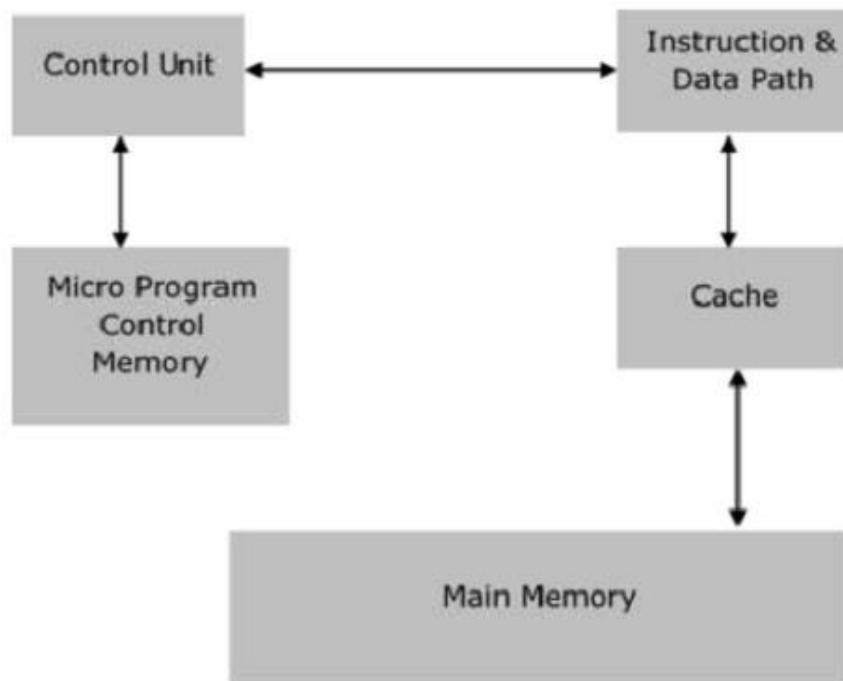
- Microprocessors are classified based on the architecture (instruction set) into RISC and CISC.
- RISC
  - It stands for Reduced Instruction Set Computer.
  - It is a microprocessor architecture that uses small instruction set of uniform length.
  - These simple instructions are executed in one clock cycle.
  - These chips are relatively simple to design.
  - They are inexpensive.
  - The disadvantage is that the computer has to repeatedly perform simple operations in order to execute a large program that has a large number of processing operations.
- Examples of RISC chips include SPARC, POWER PC.
- It has less number of instructions.
- It has fixed-length encodings for instructions.
- Simple addressing formats are supported.
- It doesn't support arrays.
- It doesn't use condition codes.
- Registers are used for procedure arguments and return addresses.
-



## CISC

- It stands for Complex Instruction Set Computer.
- This offers hundreds of instructions of different sizes to the users.
- This architecture has a set of special purpose circuits which help execute the instructions at a high speed.
- The instructions interact with memory using complex addressing modes.
- These processors reduce the size of the program.
- Consequently, they take less number of memory cycles to execute the programs.
- The overall speed of execution is high.
- Examples of CISC include Intel architecture, AMD.
- It has variable-length encodings of instructions.
- It supports array.
- It has a large number of instructions.
- Arithmetic and logical operations can be applied to memory and

# Architecture of CISC



## **Key Differences:**

1. RISC machine focuses more on software and less on hardware, whereas; CISC machine focuses more on hardware and less on software.
2. RISC machine has greater use of registers so, they use transistors for more registers, whereas; CISC machine uses a greater number of complex instructions, so they use transistors to store all their complex instructions.
3. In RISC machine, as it follows a software-based approach that is why, the code part is large, whereas; in CISC machine, as it is complex hardware driven, this makes the code part much smaller.
4. In RISC machine, due to its great and reliable software approach, an instruction can execute in a single clock cycle, whereas; in CISC machine, due to its more hardware driven approach, an instruction lags a little bit and takes more than one clock cycle.
5. The RISC instructions are quite handy and easy as they can fit in a single word, whereas; the CISC instructions are quite larger than a typical word.

BASIS	RISC	CISC
Stands for	Reduced Instruction Set Computer	Complex Instruction Set Computer
Size of instructions	Smaller and simpler instructions	Larger and Complex instructions
Execution Time	1 cycle per instruction	Multiple number of cycles per instruction
Emphasis	On Software	On hardware
Instruction Formats	Fixed (4 bytes)	Variable Length (2-6 bytes)
Control Unit	Hardwired Control Unit	Microprogrammed Control Unit
Data and Instruction Cache	Separate	Combined
Example of processors	ARM processor and Qualcomm processor are some examples	AMD, VAX and Intel x86 CPUs are some examples
CPU size	Smaller	Larger as they have larger instruction libraries

Addressing Modes	Fewer	Many
Array	Not Supported	Supported
Pipeline	Easy	Hard
Power consumption	Less	More
Referred as	Machine Oriented	Programmer Oriented
Register sets	Has multiple register sets	Has Single register set
Memory unit	Doesn't have memory unit	Have memory unit

# Cloud Security Architecture Patterns

## Cloud Security Architecture Patterns

The right pattern can help you implement security across your organization. For example, it can help you protect the CIA (confidentiality, integrity, and availability) of your cloud data assets, as well as respond to security threats. You can implement security controls directly, or use security controls as a service offered by your cloud provider or third-party vendors.

**The cloud security architecture model is usually expressed in terms of:**

- **Security controls**—which can include technologies and processes. Controls should take into account the location of each service—company, cloud provider, or third party.
- **Trust boundaries**—between the different services and components deployed on the cloud
- **Standard interfaces and security protocols**—such as SSL, IPSEC, SFTP, LDAPS, SSH, SCP, SAML, OAuth, etc.)
- **Techniques used for token management**—authentication, and authorization
- **Encryption methods** including algorithms like 128-bit AES, Triple DES, RSA, Blowfish.
- **Security event logging**—ensuring all relevant security events are captured, prioritized, and delivered to security teams.

Here are a few best practices that you can follow to enhance the security of your cloud-based assets:

### **Enforce policies and data governance:**

It is entirely the enterprise's responsibility to put in place and enforce policies for cloud data ownership and responsibility. At the most basic level, the enterprise must understand and classify its data so that the appropriate security measures can be implemented according to the varying levels of data sensitivity.

### **Diligently manage identity and access controls:**

Identity and access management (IAM) in the cloud is substantially more complex than it is in closed, monolithic environments. Cloud providers offer best practice guidelines as well as tools and managed services to help organizations handle IAM, but it's up to the organization to use them effectively.

# Security Control

Each security control should be clearly defined using the following attributes:

- **Service function**—what is the service’s role? For example, encryption, authorization, event data collection.
- **Logical location**—public cloud service, third party service, or on-premises. Location affects performance, availability, firewall policies, and service management.
- **Protocol**—what protocol is used to access the service? For example, REST, HTTPS, SSH.
- **Input/Output** – what does the service receive and what is it expected to deliver? For example, input is a JSON feed and output is the same feed with encrypted payload data.
- **Control mechanisms**—what types of control does the service achieve? For example, data at rest protection, user authentication, application authentication.
- **Users and operators**—who operates or benefits from the service? For example, endpoint devices, end users, business managers, security analysts.

# IaaS Cloud Computing Security Architecture

## IaaS Cloud Computing Security Architecture

In an IaaS framework, the cloud provider is completely responsible for the physical resources and shares responsibility with the customer for the security of the host infrastructure and network; all the rest is the responsibility of the customer. This level brings the customer the most freedom, but also places the majority of the responsibility in their hands. The principles of IaaS closely follow the shared responsibility model for providers like AWS and Azure.

IaaS provides storage and network resources in the cloud. It relies heavily on APIs to help manage and operate the cloud. However, cloud APIs are often not secure, because they are open and easily accessible from the web.

The cloud service provider (CSP) is responsible for securing the infrastructure and abstraction layer used to access the resources. Your organization's security obligations cover the rest of the layers, mainly containing the business applications.

To better visualize cloud network security issues, deploy a Network Packet Broker (NPB) in an IaaS environment. The NPB sends traffic and data to a Network Performance Management (NPM) system, and to the relevant security tools. In addition, establish logging of events occurring on network endpoints.

IaaS cloud deployments require the following additional security features:

- Network segmentation
- Intrusion Detection System and Intrusion Prevention System (IDS/IPS)
- Virtual firewalls placed in front of web applications to protect against malicious code, and at the edge of the cloud network
- Virtual routers

# PaaS Cloud Computing Security Architecture

## **PaaS Cloud Computing Security Architecture**

In a PaaS framework, the provider also takes full responsibility for hosting physical infrastructure and network security, but it also shares responsibility with the customer at the application and access control levels.

Application software, virtual machines and instances, and services such as AWS Elastic Beanstalk and AWS Lambda typically fall into the Platform-as-a-Service category.

PaaS platforms enable organizations to build applications without the overhead and complexity associated with managing hardware and back-end software. In a PaaS model, the CSP protects most of the environment. However, the company is still responsible for the security of the applications it is developing.

Therefore, a PaaS security architecture is similar to a SaaS model. Ensure you have CASP, logging and alerting, IP restrictions and an API gateway to ensure secure internal

# SaaS Cloud Computing Security Architecture

## **SaaS Cloud Computing Security Architecture**

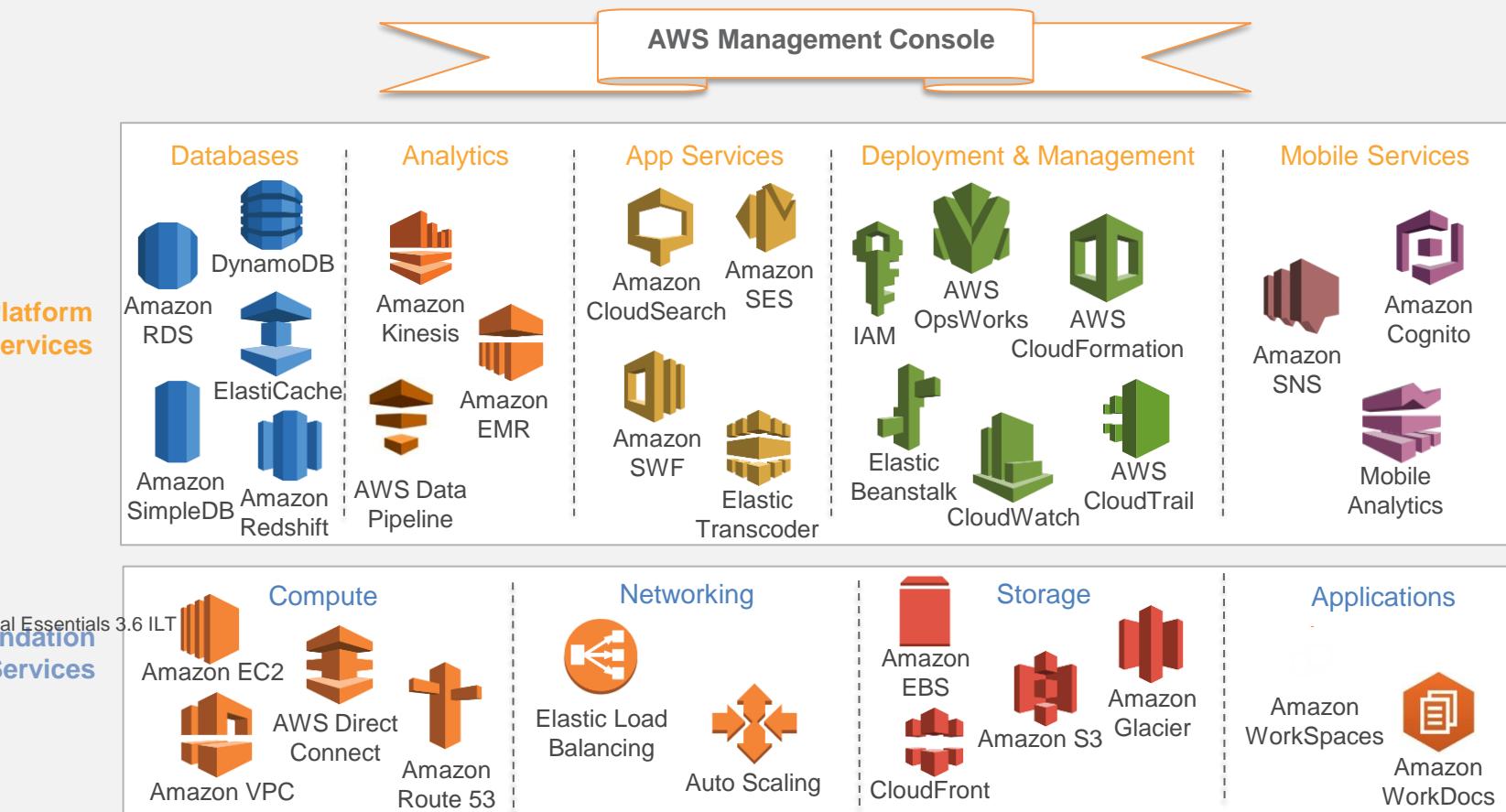
SaaS framework, the provider takes full responsibility for application controls while sharing responsibility with the customer for access control as well as client/endpoint protection. Generally, SaaS companies provide business applications or other consumer apps over the internet that may run in the cloud,

SaaS services provide access to software applications and data through a browser. The specific terms of security responsibility may vary between services, and are sometimes up for negotiation with the service provider.

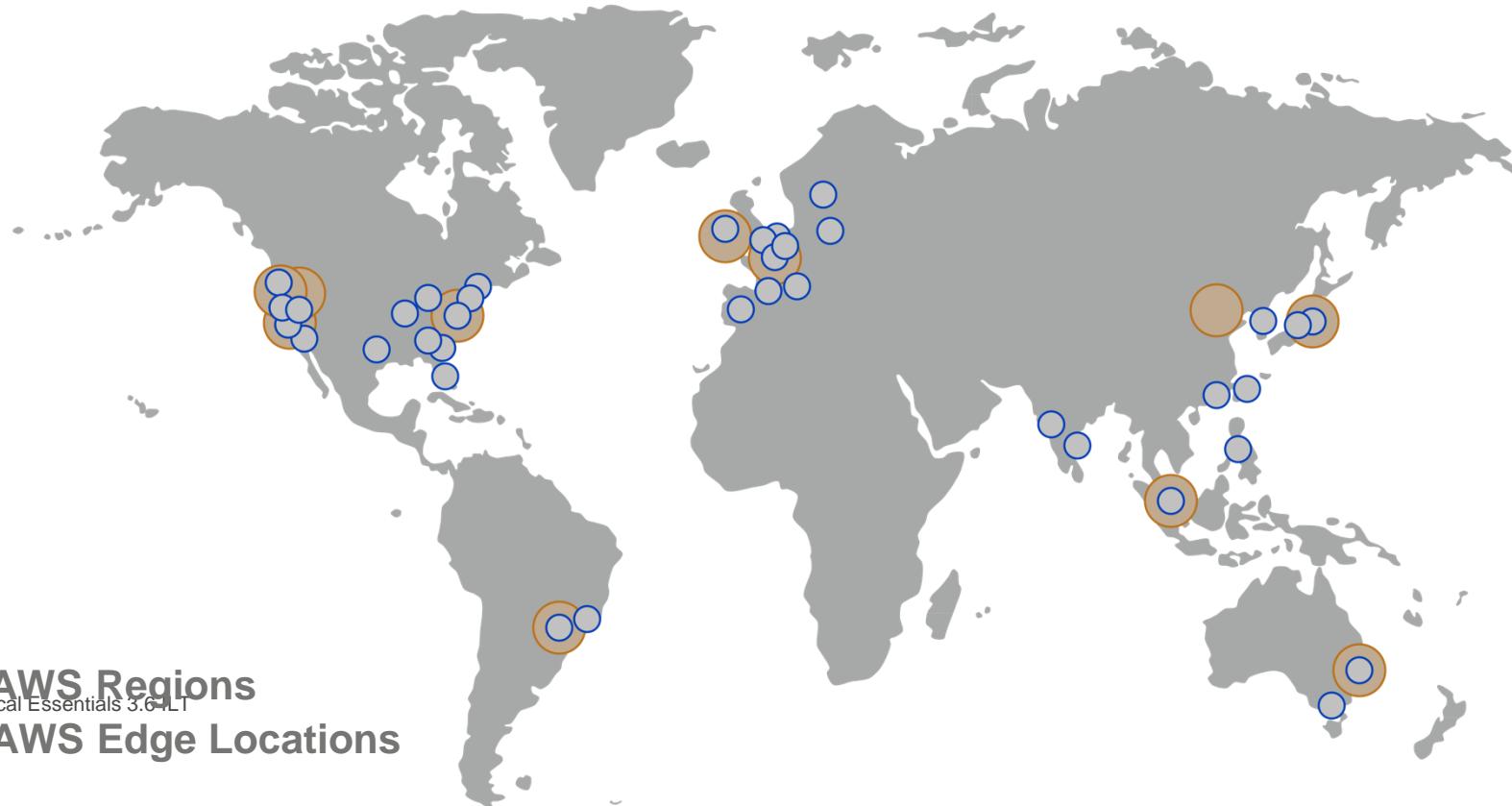
Cloud Access Security Brokers (CASB) offers logging, auditing, access control and encryption capabilities that can be critical when investigating security issues in a SaaS product. In addition, make sure your SaaS environment has:

- Logging and alerting
- IP whitelists and/or blacklists
- API gateways, in case the service is accessed via API

# AWS Services

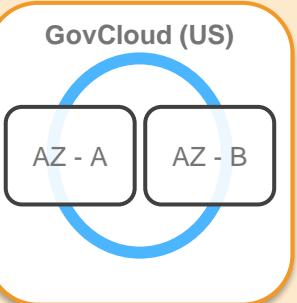
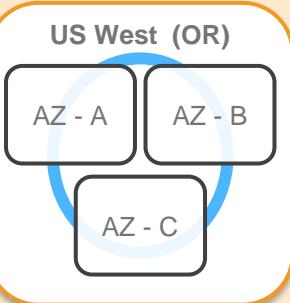
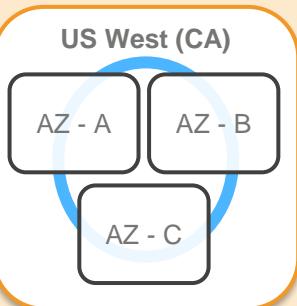
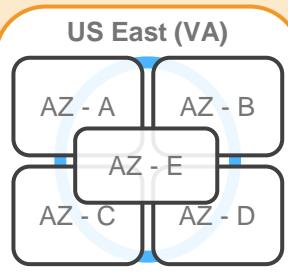


# Regions and Edge Locations

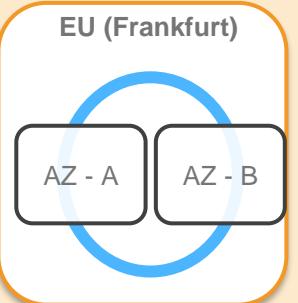
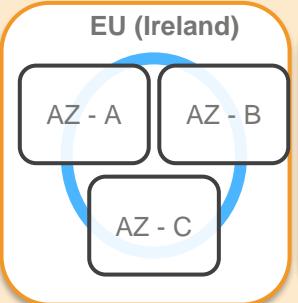
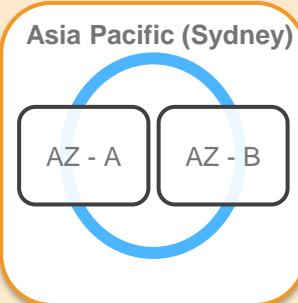
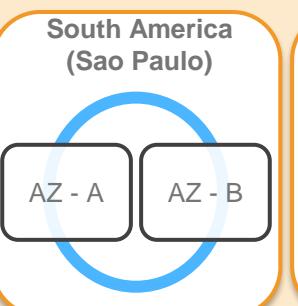
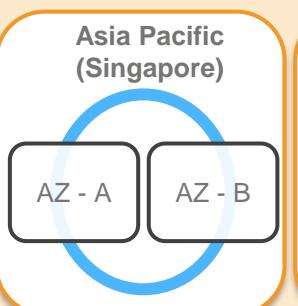
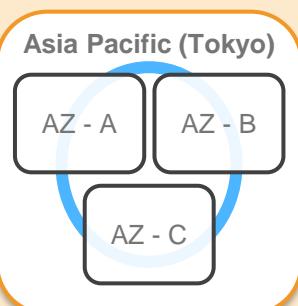


# AWS Regions and Availability Zones (AZ)

## US Regions

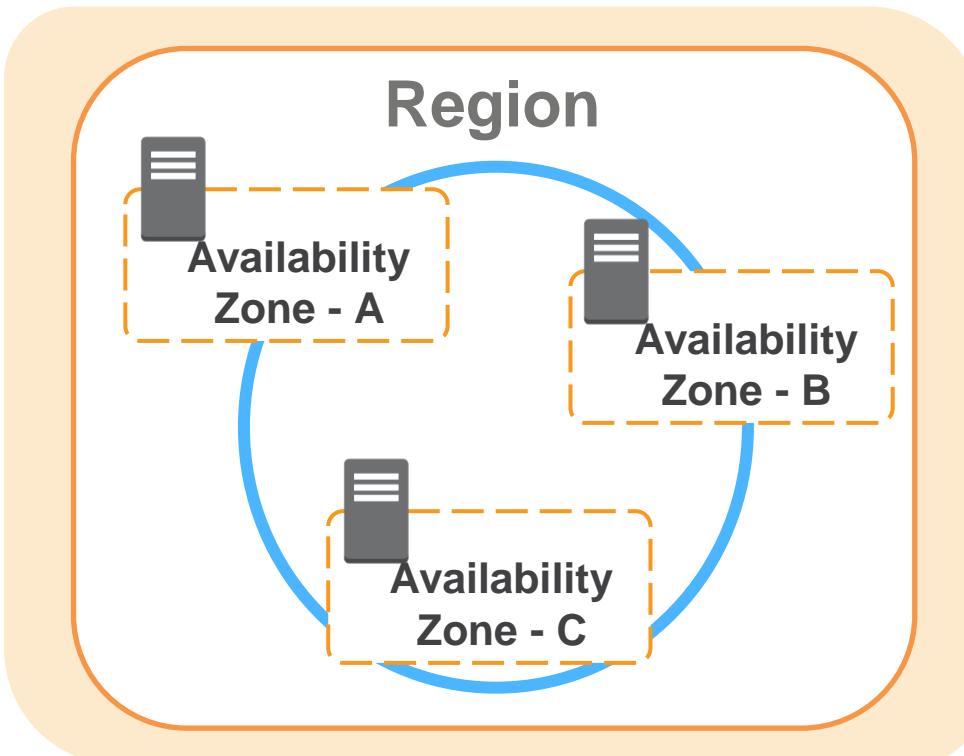


## Global Regions



Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

# Achieving High Availability Using Multi-AZ



# Shared Security Responsibility—AWS

Customer

AWS

**Customer Data**

**Platform, Applications, Identity & Access Management**

**Operating System, Network & Firewall Configuration**

Client-side Data Encryption &  
Data Integrity Authentication

Server-side Encryption  
(File System and/or Data)

Network Traffic Protection  
(Encryption/Integrity/Identity)

**Foundation Services**

**Compute**

**Storage**

**Database**

**Network**

**AWS Global  
Infrastructure**

**Availability Zones**

**Regions**

**Edge  
Locations**

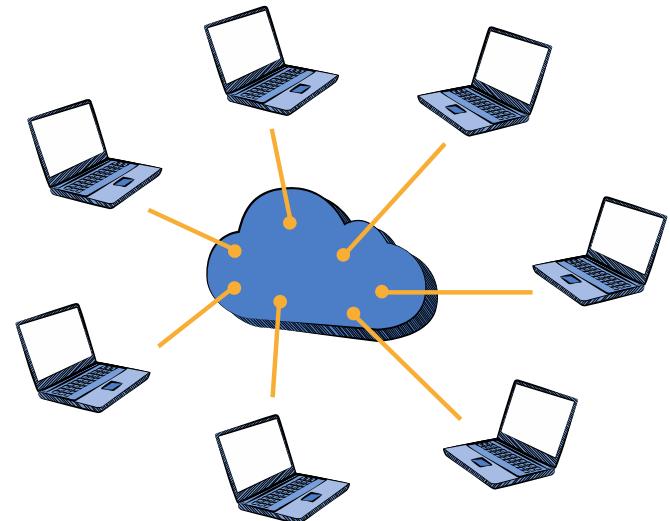
# Physical Security

- 24/7 trained security guards
- Locations in nondescript, undisclosed facilities
- Two-factor authentication for ingress
- Authorization for datacenter access



# Hardware, Software, and Network

- Automated change-control process
- Bastion servers that record all access attempts
- Firewall and other boundary devices
- AWS monitoring tools



# SSL Endpoints

## SSL Endpoints

### Secure Transmission

Establish secure communication sessions (HTTPS) using SSL

## Security Groups

### Instance Firewalls

Configure firewall rules for instances using Security Groups

## IAM

### User Accounts

Create individual IAM accounts so that each of your users has their own AWS security credentials

## VPC

### Subnet Control

In your Virtual Private Cloud, create low-level networking constraints for resource access, such as public and private subnets, internet gateways, and NATs

# Security Groups

## SSL Endpoints

### Secure Transmission

Establish secure communication sessions (HTTPS) using SSL

## Security Groups

### Instance Firewalls

Configure firewall rules for instances using Security Groups

## IAM

### User Accounts

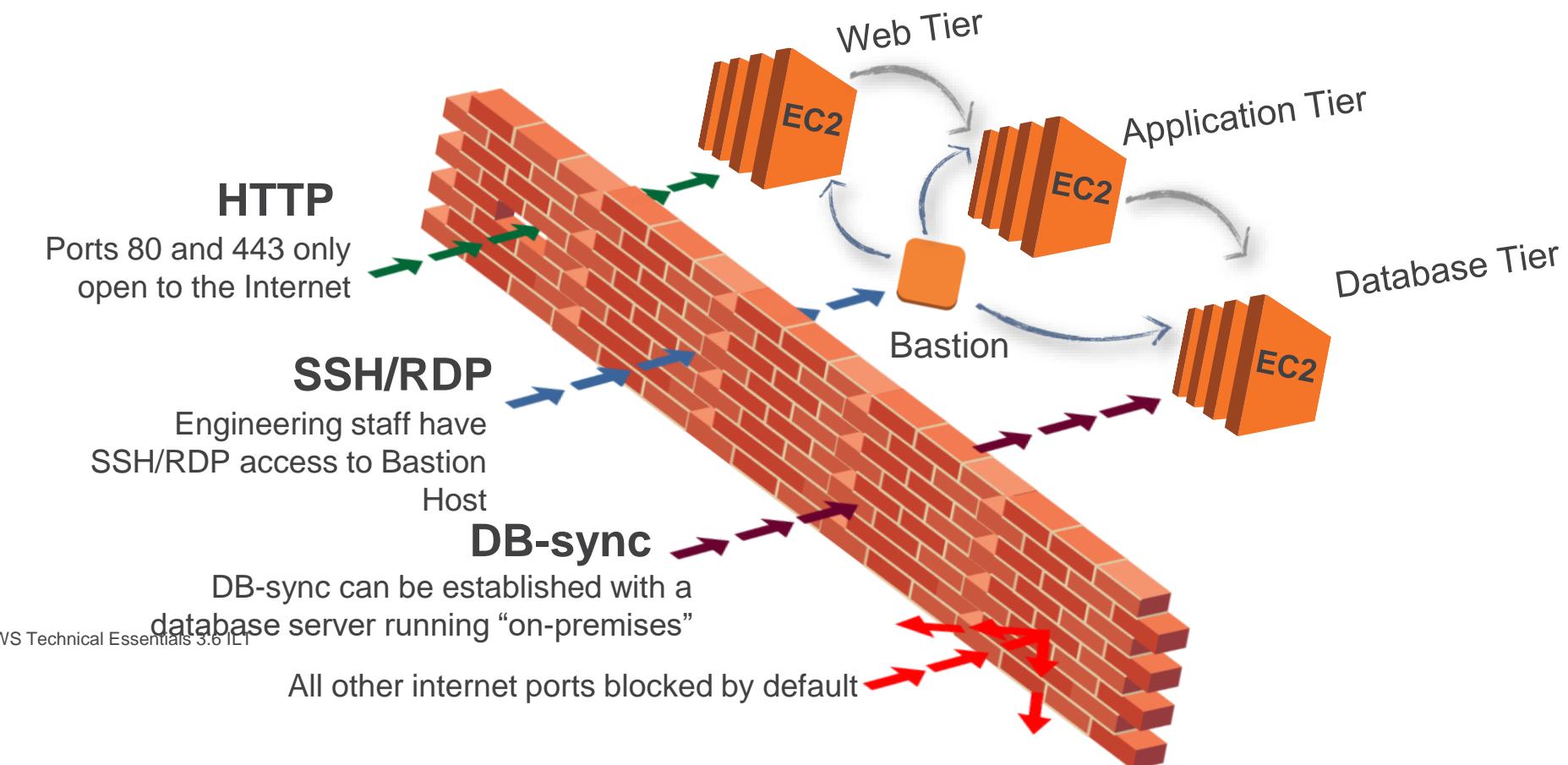
Create individual IAM accounts so that each of your users has their own AWS security credentials

## VPC

### Subnet Control

In your Virtual Private Cloud, create low-level networking constraints for resource access, such as public and private subnets, internet gateways, and NATs

# AWS Multi-tier Security Groups



# AWS Identity and Access Management (IAM)

SSL Endpoints	Security Groups	IAM	VPC
<p><b>Secure Transmission</b></p> <p>Establish secure communication sessions (HTTPS) using SSL</p>	<p><b>Instance Firewalls</b></p> <p>Configure firewall rules for instances using Security Groups</p>	<p><b>User Accounts</b></p> <p>Create individual IAM accounts so that each of your users has their own AWS security credentials</p>	<p><b>Subnet Control</b></p> <p>In your Virtual Private Cloud, create low-level networking constraints for resource access, such as public and private subnets, internet gateways, and NATs</p>

# Account Control with IAM

- AWS Identity and Access Management (IAM):
  - Securely control access to AWS services and resources
  - Create and manage AWS users and groups
- AWS Master accounts should not be used for production systems!!
  - You should be using IAM user accounts.



# Amazon Virtual Private Cloud (VPC)

## SSL Endpoints

### Secure Transmission

Establish secure communication sessions (HTTPS) using SSL

## Security Groups

### Instance Firewalls

Configure firewall rules for instances using Security Groups

## IAM

### User Accounts

Create individual IAM accounts so that each of your users has their own AWS security credentials

## VPC

### Subnet Control

In your Virtual Private Cloud, create low-level networking constraints for resource access, such as public and private subnets, internet gateways, and NATs

# EC2 Feature Consideration

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud.

Amazon EC2 provides a number of powerful features for building scalable, failure resilient, enterprise class applications

- Elastic Web-Scale Computing
- Completely Controlled
- Flexible Cloud Hosting Services
- Designed for use with other Amazon Web Services
- Reliable
- Secure
- Inexpensive
- Multi Locations
- Elastic IP address
- Amazon Virtual Private Cloud
- Auto Scaling
- Amazon Cloud Watch
- Enhanced Networking
- EC2 video link <https://www.youtube.com/watch?v=wNr7YqjjzOY>

# EC2 Elastic Properties

- Elastic – Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days.
- You can commission one, hundreds or even thousands of server instances simultaneously.
- controlled with web service APIs, application can automatically scale itself up and down depending on its needs.
- Elastic Block Store vs. local Disk (not backup)
- Elastic IP Addresses vs. Static IP Addresses
- Interesting charging scheme; you are charged when not using it
- programmatically remapping your public IP addresses to any instance in your account

# EC2 Instance Type

Instance Type	Instance Name	Application Suitability
General Purpose	T2,M3, M4	Development environments, build servers, code repositories, low-traffic websites and web applications, micro services, early product experiments, small databases
Compute Optimized	C3, C4	High performance front-end fleets, web-servers, batch processing, distributed analytics, high performance science and engineering applications, ad serving, MMO gaming, and video-encoding
Memory Optimized	R3	We recommend memory-optimized instances for high performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, larger deployments of SAP, Microsoft SharePoint, and other enterprise applications.
GPU	G2	3D application streaming, machine learning, video encoding, and other server-side graphics or GPU compute workload
Storage Optimized	I2	NOSQL Databases like Cassandra and MongoDB, scale out transactional databases, data warehousing, Hadoop, and cluster file systems.
Dense Storage	D2	Massively Parallel Processing (MPP) data warehousing, MapReduce and Hadoop distributed computing, distributed file systems, network file systems, log or data-processing applications

AW

# AWS EC2 Pricing Model

- Free Usage Tier
- On-Demand Instances
  - Start and stop instances whenever you like, costs are rounded up to the nearest hour. (Worst price)
- Reserved Instances
  - Pay up front for one/three years in advance. (Best price)
  - Unused instances can be sold on a secondary market.
- Spot Instances
  - Specify the price you are willing to pay, and instances get started and stopped without any warning as the marked changes.

# AWS EC2 Free UsageTier

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

# Storage

- Instance –store : disappears with the instance (transient)
- Block storage: SAN-like, persists across time
- S3 is Object storage independent of an instance
- Glacier for archival purposes store it now and retrieve it at a later date
- Amazon: SimpleDB: Relational database better than MySQL or Oracle for reliability.

# Elastic Block Storage (EBS)

- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a file system on a running EC2 instance where it can be **updated incrementally**. Unlike an instance store, an EBS volume is **persistent**.
- (Compare to an S3 object, which is essentially a file that must be accessed in its entirety.)
- Amazon EBS is particularly well-suited for use as the primary storage for a database or file system, or for any applications that require access to raw block-level storage
- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

# Simple Storage Service (S3)

- Simple Storage service is a storage for internet
- The number of objects you can store in S3 is unlimited
- A **bucket** is a container for objects and describes location, logging, accounting, and access control. A bucket can hold any number of **objects**, which are files of up to 5TB. A bucket has a name that must be **globally unique**.
- Fundamental operations corresponding to HTTP actions:
  - `http://bucket.s3.amazonaws.com/object`
  - POST a new object or update an existing object.
  - GET an existing object from a bucket.
  - DELETE an object from the bucket
  - LIST keys present in a bucket, with a filter.
- A bucket has a **flat directory structure** (despite the appearance given by the interactive web interface.)
- Amazon S3 works well for fast growing websites hosting data intensive, user-generated content, such as video and photo sharing sites.
- No set-up fee, No monthly minimum
- Video link <https://www.youtube.com/watch?v=1qrjFb0ZTm8>

# S3 Bucket Naming

- Flat namespace
- Names may contain only lowercase letters, numbers, periods, underscores, and dashes, and must start with a number or letter
- Create your own namespace with your own buckets

# Simple Storage Services (S3)

## Bucket Properties

- Versioning – If enabled, POST/DELETE result in the creation of new versions without destroying the old.
- Lifecycle – Delete or archive objects in a bucket a certain time after creation or last access or number of versions.
- Access Policy – Control **when and where** objects can be accessed.
- Access Control – Control who **may** access objects in this bucket.
- Logging – Keep track of how objects are accessed.
- Notification – Be notified when failures occur.

# Durability

- Amazon claims about S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains.
  - 99.99999999% durability of objects over a given year.
- Amazon claims about EBS:
  - Amazon EBS volume data is replicated across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component.
  - Volumes <20GB modified data since last snapshot have an annual failure rate of 0.1% - 0.5%, resulting in complete loss of the volume.
  - Commodity hard disks have an AFR of about 4%.
- Amazon claims about Glacier is the same as S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains PLUS periodic internal integrity checks.
  - 99.99999999% durability of objects over a given year.

# AWS Auto Scaling Capability

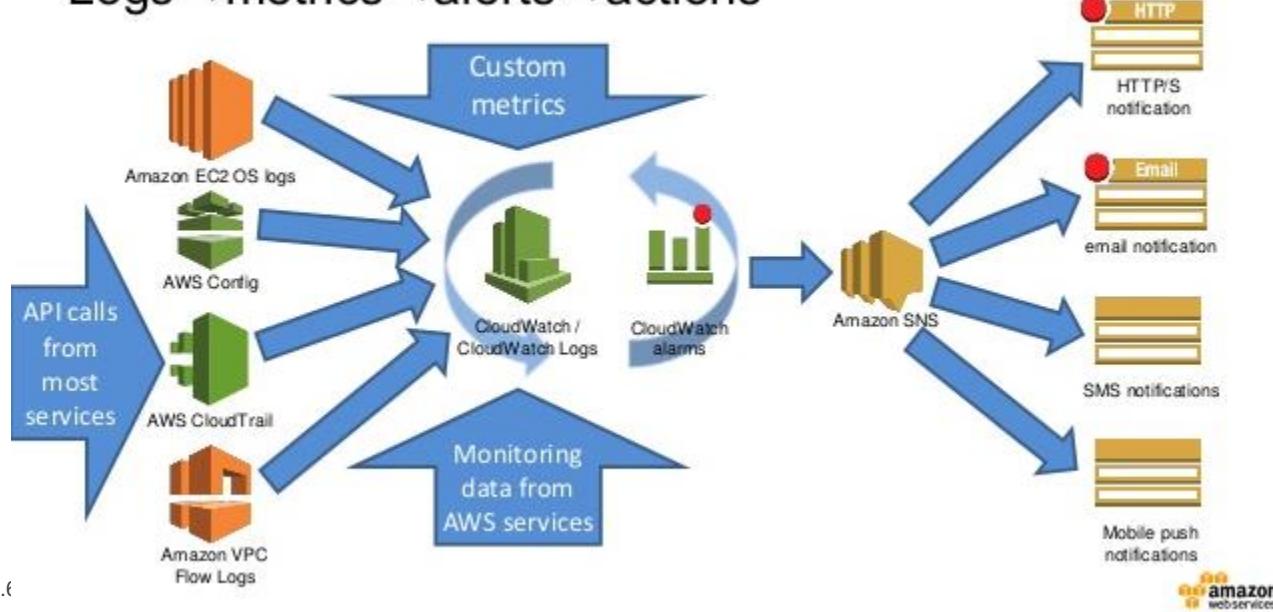
- Auto Scaling helps you maintain application availability and allows you to scale your [Amazon EC2](#) capacity up or down automatically according to conditions you define
- Manage unhealthy EC2 compute instances
- Ensure minimum number instances are always running
- Launched new instances in event of failure or performance degradation (assume 30-120 seconds in most conditions)
- Seamlessly attach auto scaled compute instances to load balancer (ELB)
- Video Link <https://www.youtube.com/watch?v=7SfVZqOVcCI>

# AWS Elastic Load Balancer

- AWS ELB provides load balancing service with thousands of EC2 servers behind them
- AWS ELB will automatically Scale up /down the load balancing servers in backend
- The theoretical maximum response rate of AWS ELB is limitless
- It can handle 20,000+ concurrent requests easily

# AWS Cloud Watch

Logs→metrics→alerts→actions



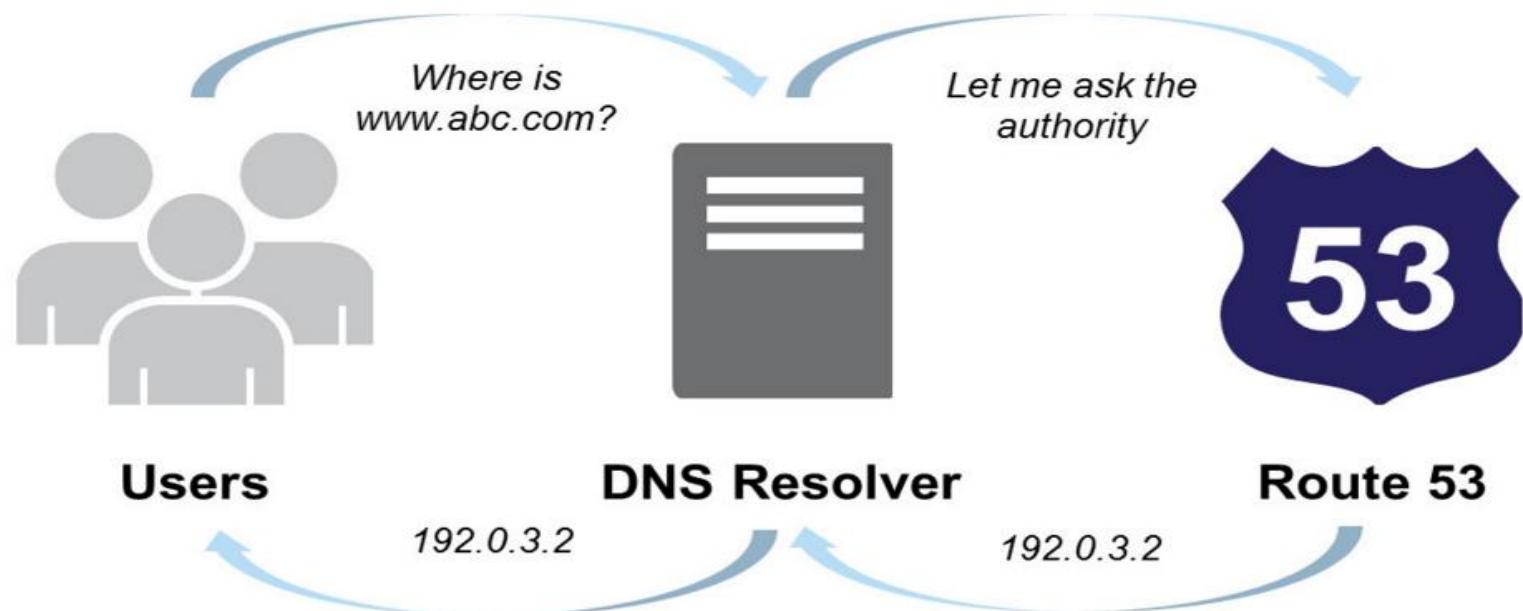
# AWS Cloud Front

- Cloud-based content distributing network enables you to place the content at the edges of the network for rapid delivery.
- Place the contents in S3 and run the application from anywhere and the content is moved to where the application is (to the edges).
- Video Link

AWS Technical Essentials 3.6 ILT

[https://www.youtube.com/watch?v=\\_oTj\\_3o1ceE](https://www.youtube.com/watch?v=_oTj_3o1ceE)

# AWS Route 53



# Amazon Relational Database Service (RDS)

- Amazon Relational Database Service a web service that provides the capabilities of MySQL, Oracle, or Microsoft SQL Server relational database as a managed, cloud-based service
  - On-demand provisioning
  - No operating system for you to access
  - Platforms:
    - MySQL
    - Oracle
    - SQL Server
    - PostgreSQL
  - “Mostly” pre-configured
  - Basic monitoring / metrics provided
  - Automated backups
  - Automated recovery
  - User initiated “snapshots”
- AWS•Technique  
Automated replication
- Software upgrades provided

# RDS Features and Functionality

- Pre-configured Parameters
- Monitoring and Metrics
- Automatic Software Patching
- Automated Backups
- DB Snapshots
- Push-Button Scaling
- Automatic Host Replacement
- **Replication:** two features

AWS Technical Essentials 3.6 ILT

Multi-AZ Deployment, Read Replica

# Multi-AZ Deployment

- **Availability Zones:** are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones.
- Run a DB Instance as a **Multi-AZ deployment**, the “primary” serves database writes and reads. Amazon RDS provisions and maintains a “standby” behind the scenes, which is an up-to-date replica of the primary. The standby is “promoted” in failover scenarios. After failover, the standby becomes the primary and accepts your database operations.

# Oracle Deployment options

## RDS

- Quick provisioning
- “Easy” management
- Simple environment setup
- Simplified replication strategy
- No OS-level control
- Limited granular fine tuning
- Limited platform / versions

## EC2+Database

- You manage it yourself
- OS & storage overhead
- Software / version management
- Configuration
- Usually more costly
- More control over config / performance
- Allows more complex setups

# Scalability- Scale Up vs. Scale Out

- Scale Up/Vertical Scaling
  - Select next available configuration (EC2, RDS)
  - Relatively simple but limited scalability
- Scale Out/Horizontal Scaling
  - Add additional resources
  - Complicated but high scalability
  - RDS Read replicas

# RDS Limitation

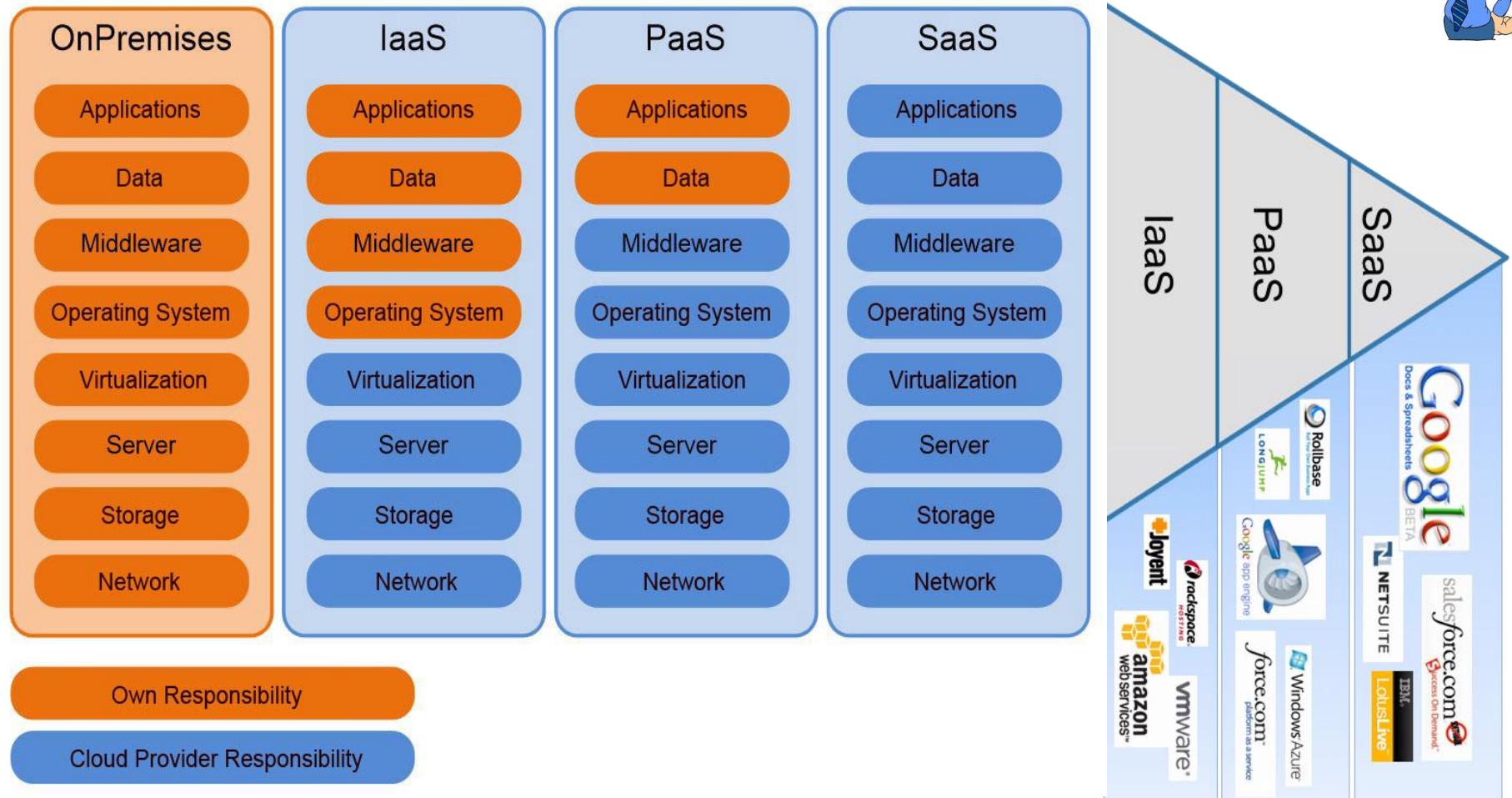
- Failovers are NOT instant.
    - Can take up to 6 minutes. Maybe more, depending on database size
  - Only limited platforms / versions are supported.
  - Upgrades / patching requires downtime.
    - You set an allowable maintenance window. Amazon will upgrade sometime during that window.
  - Can suffer from “noisy neighbor” syndrome.
  - Performance issues are sometimes hard to pinpoint
- Master-Master replication is NOT supported



# heard of 3 models of Cloud Computing?



# Yes, Yes, IaaS, PaaS and SaaS





**BITS** Pilani  
Pilani Campus

# BITS Pilani presentation

Mridul Moitra  
Cloud Computing





<CSI ZG527 / SS ZG527 / SE ZG527  
Cloud Computing  
Lecture No. 3

**BITS** Pilani



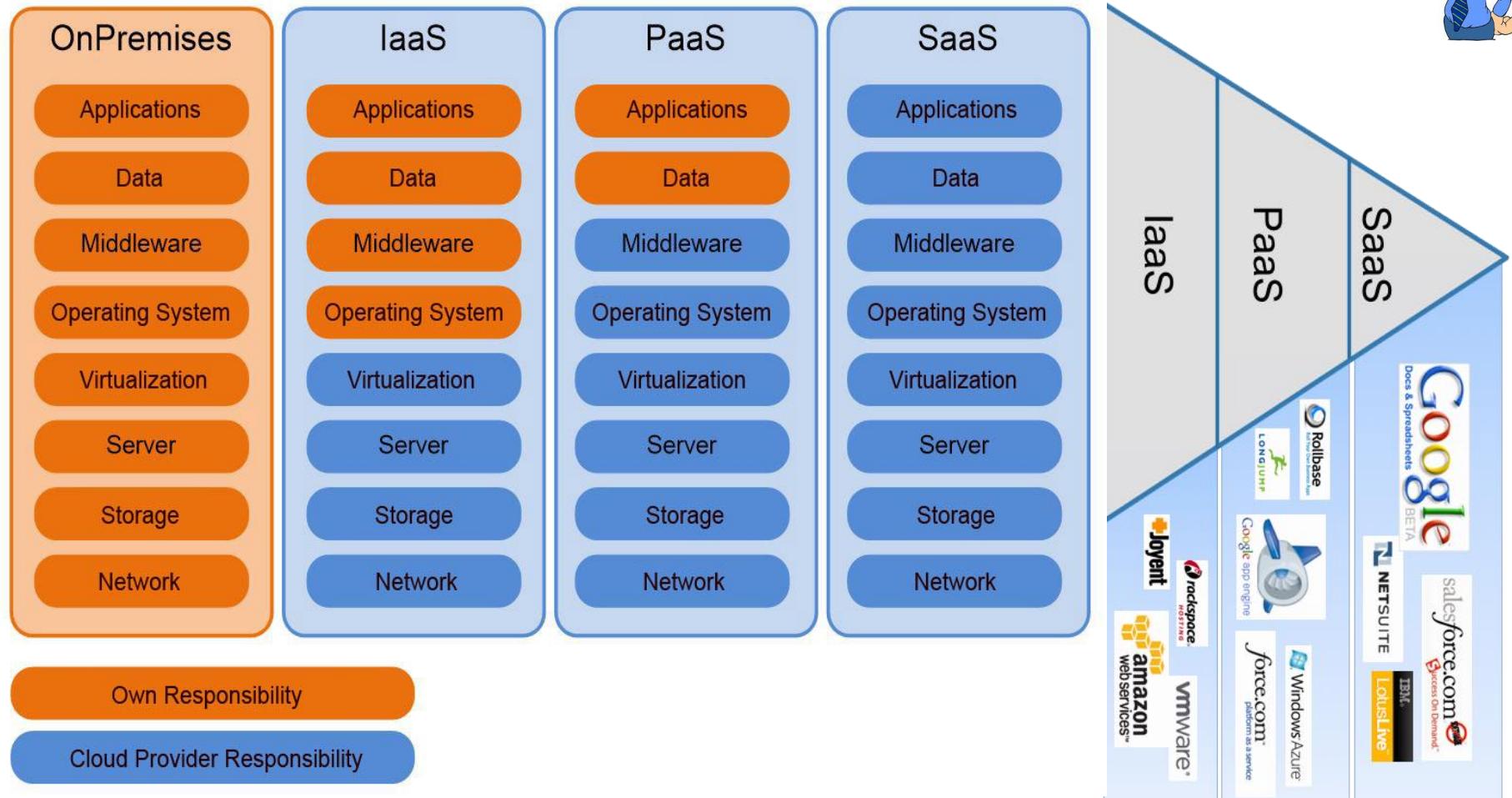
# IaaS



# 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

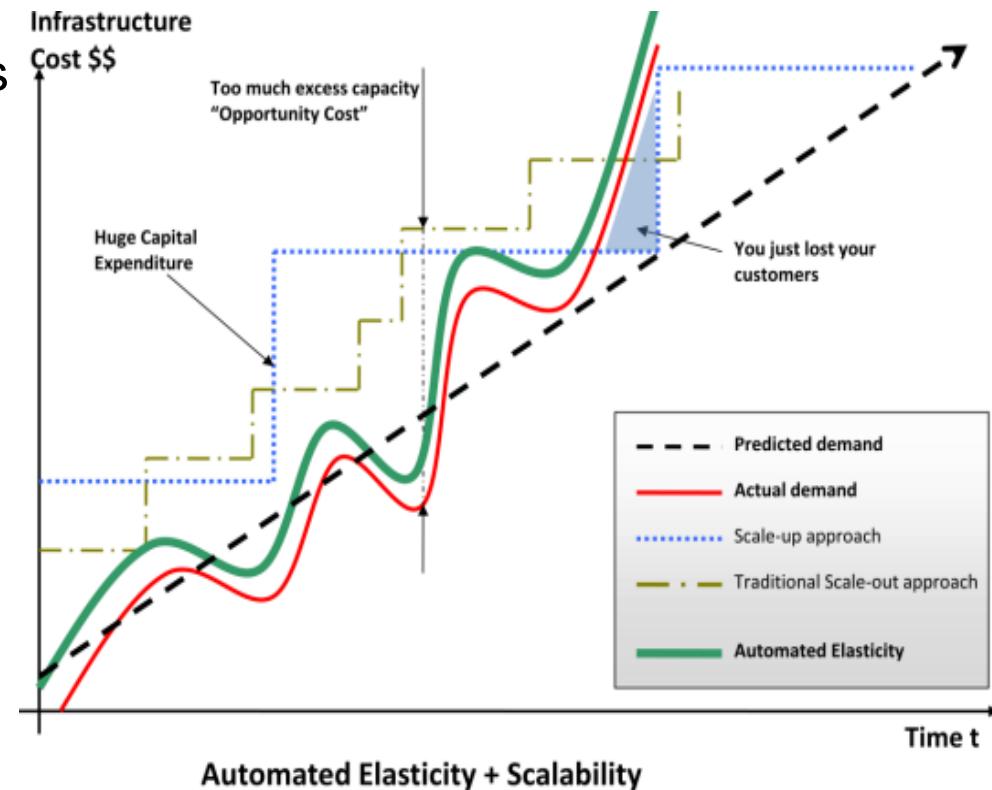
## Elasticity:

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”

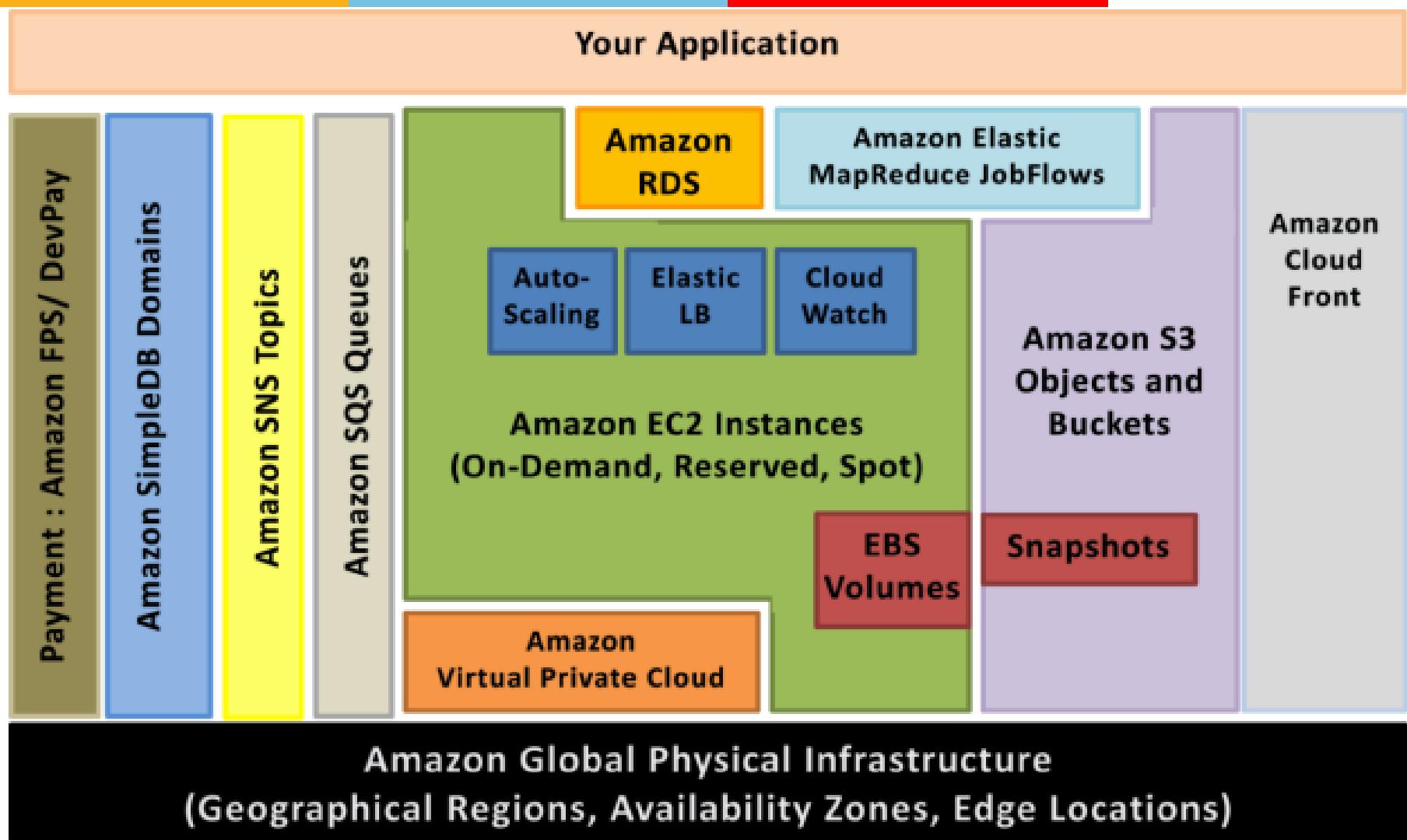
OR simply put “Ability of a system to **expand or contract** its dedicated resources to meet the demand”

&

## Virtualization



# AWS infrastructure services



# EC2 Elastic Properties

- Elastic – Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days.
- You can commission one, hundreds or even thousands of server instances simultaneously.
- controlled with web service APIs, application can automatically scale itself up and down depending on its needs.
- Elastic Block Store vs. local Disk (not backup)
- Elastic IP Addresses vs. Static IP Addresses
- Interesting charging scheme; you are charged when not using it
- programmatically remapping your public IP addresses to any instance in your account

# EC2 Instance Type

Instance Type	Instance Name	Application Suitability
General Purpose	T2,M3, M4	Development environments, build servers, code repositories, low-traffic websites and web applications, micro services, early product experiments, small databases
Compute Optimized	C3, C4	High performance front-end fleets, web-servers, batch processing, distributed analytics, high performance science and engineering applications, ad serving, MMO gaming, and video-encoding
Memory Optimized	R3	We recommend memory-optimized instances for high performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, larger deployments of SAP, Microsoft SharePoint, and other enterprise applications.
GPU	G2	3D application streaming, machine learning, video encoding, and other server-side graphics or GPU compute workload
Storage Optimized	I2	NOSQL Databases like Cassandra and MongoDB, scale out transactional databases, data warehousing, Hadoop, and cluster file systems.
Dense Storage	D2	Massively Parallel Processing (MPP) data warehousing, MapReduce and Hadoop distributed computing, distributed file systems, network file systems, log or data-processing applications

AW

# AWS EC2 Pricing Model

- Free Usage Tier
- On-Demand Instances
  - Start and stop instances whenever you like, costs are rounded up to the nearest hour. (Worst price)
- Reserved Instances
  - Pay up front for one/three years in advance. (Best price)
  - Unused instances can be sold on a secondary market.
- Spot Instances
  - Specify the price you are willing to pay, and instances get started and stopped without any warning as the marked changes.

# AWS EC2 Free UsageTier

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

# Storage

- Instance store : disappears with the instance (transient)
- Block storage: SAN-like, persists across time
- S3 is Object storage independent of an instance
- Glacier for archival purposes store it now and retrieve it at a later date

# Elastic Block Storage (EBS)

- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a file system on a running EC2 instance where it can be **updated incrementally**. Unlike an instance store, an EBS volume is **persistent**.
- (Compare to an S3 object, which is essentially a file that must be accessed in its entirety.)
- Amazon EBS is particularly well-suited for use as the primary storage for a database or file system, or for any applications that require access to raw block-level storage
- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

# Simple Storage Service (S3)

- Simple Storage service is a storage for internet
- The number of objects you can store in S3 is unlimited
- A **bucket** is a container for objects and describes location, logging, accounting, and access control. A bucket can hold any number of **objects**, which are files of up to 5TB. A bucket has a name that must be **globally unique**.
- Fundamental operations corresponding to HTTP actions:
  - `http://bucket.s3.amazonaws.com/object`
  - POST a new object or update an existing object.
  - GET an existing object from a bucket.
  - DELETE an object from the bucket
  - LIST keys present in a bucket, with a filter.
- A bucket has a **flat directory structure** (despite the appearance given by the interactive web interface.)
- Amazon S3 works well for fast growing websites hosting data intensive, user-generated content, such as video and photo sharing sites.
- No set-up fee, No monthly minimum
- Video link <https://www.youtube.com/watch?v=1qrjFb0ZTm8>

# S3 Bucket Naming

- Flat namespace
- Names may contain only lowercase letters, numbers, periods, underscores, and dashes, and must start with a number or letter
- Create your own namespace with your own buckets

# Simple Storage Services (S3)

## Bucket Properties

- Versioning – If enabled, POST/DELETE result in the creation of new versions without destroying the old.
- Lifecycle – Delete or archive objects in a bucket a certain time after creation or last access or number of versions.
- Access Policy – Control **when and where** objects can be accessed.
- Access Control – Control who **may** access objects in this bucket.
- Logging – Keep track of how objects are accessed.
- Notification – Be notified when failures occur.

# Durability

- Amazon claims about S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains.
  - 99.99999999% durability of objects over a given year.
- Amazon claims about EBS:
  - Amazon EBS volume data is replicated across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component.
- Amazon claims about Glacier is the same as S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains PLUS periodic internal integrity checks.
  - 99.99999999% durability of objects over a given year.

# AWS Auto Scaling Capability

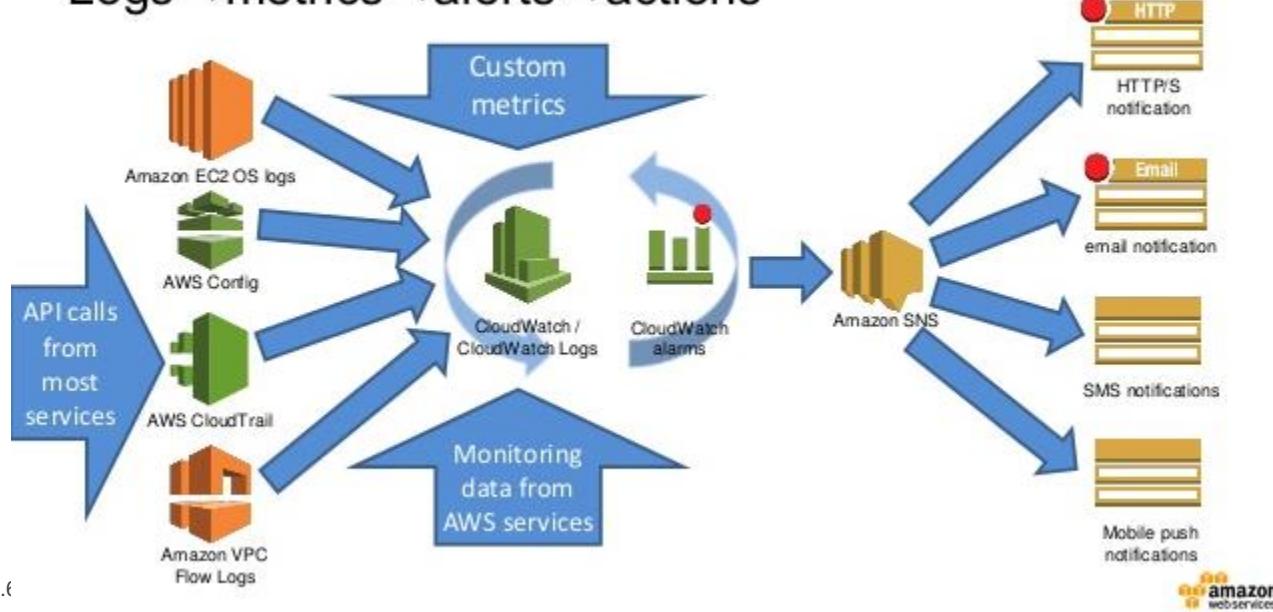
- Auto Scaling helps you maintain application availability and allows you to scale your [Amazon EC2](#) capacity up or down automatically according to conditions you define
- Manage unhealthy EC2 compute instances
- Ensure minimum number instances are always running
- Launched new instances in event of failure or performance degradation (assume 30-120 seconds in most conditions)
- Seamlessly attach auto scaled compute instances to load balancer (ELB)

# AWS Elastic Load Balancer

- AWS ELB provides load balancing service with thousands of EC2 servers behind them
- AWS ELB will automatically Scale up /down the load balancing servers in backend
- The theoretical maximum response rate of AWS ELB is limitless
- It can handle 20,000+ concurrent requests easily

# AWS Cloud Watch

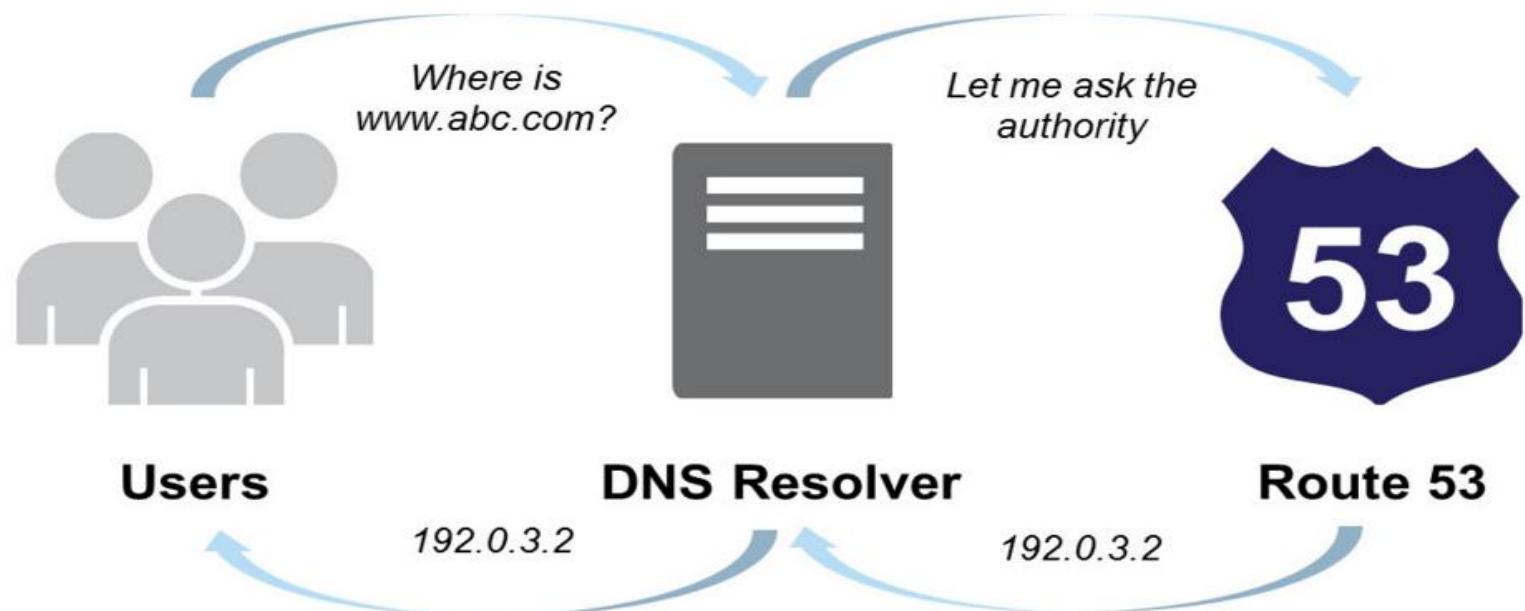
Logs→metrics→alerts→actions



# AWS Cloud Front

- Cloud-based content distributing network enables you to place the content at the edges of the network for rapid delivery.
- Place the contents in S3 and run the application from anywhere and the content is moved to where the application is (to the edges).

# AWS Route 53



# Multi-AZ Deployment

- **Availability Zones:** are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones.
- Run a DB Instance as a **Multi-AZ deployment**, the “primary” serves database writes and reads. Amazon RDS provisions and maintains a “standby” behind the scenes, which is an up-to-date replica of the primary. The standby is “promoted” in failover scenarios. After failover, the standby becomes the primary and accepts your database operations.

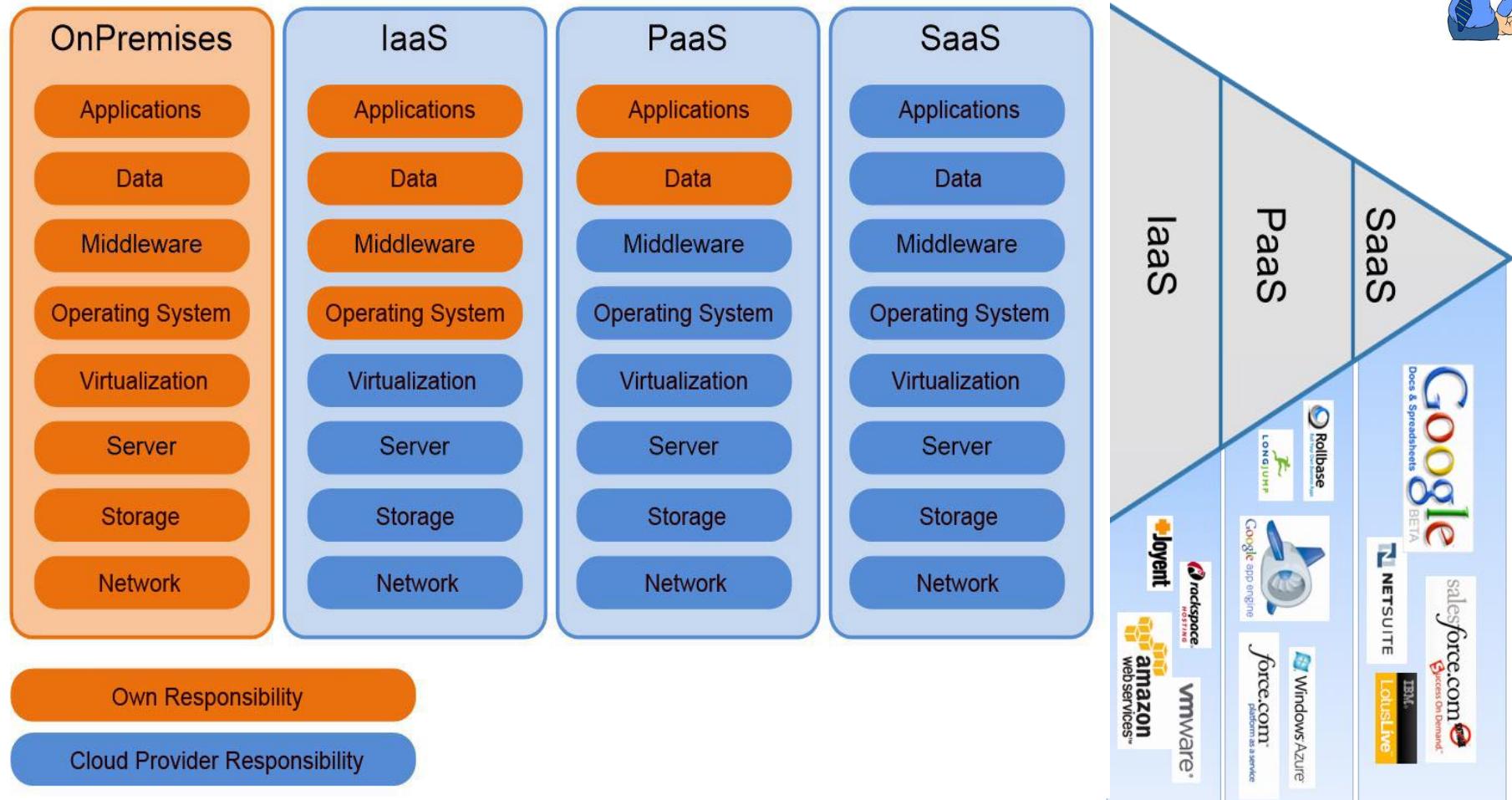
# Scalability- Scale Up vs. Scale Out

- Scale Up/Vertical Scaling
  - Select next available configuration (EC2, RDS)
  - Relatively simple but limited scalability
- Scale Out/Horizontal Scaling
  - Add additional resources
  - Complicated but high scalability
  - RDS Read replicas



# heard of 3 models of Cloud Computing?

# Yes, Yes, IaaS, PaaS and SaaS



# Introduction to PaaS

---

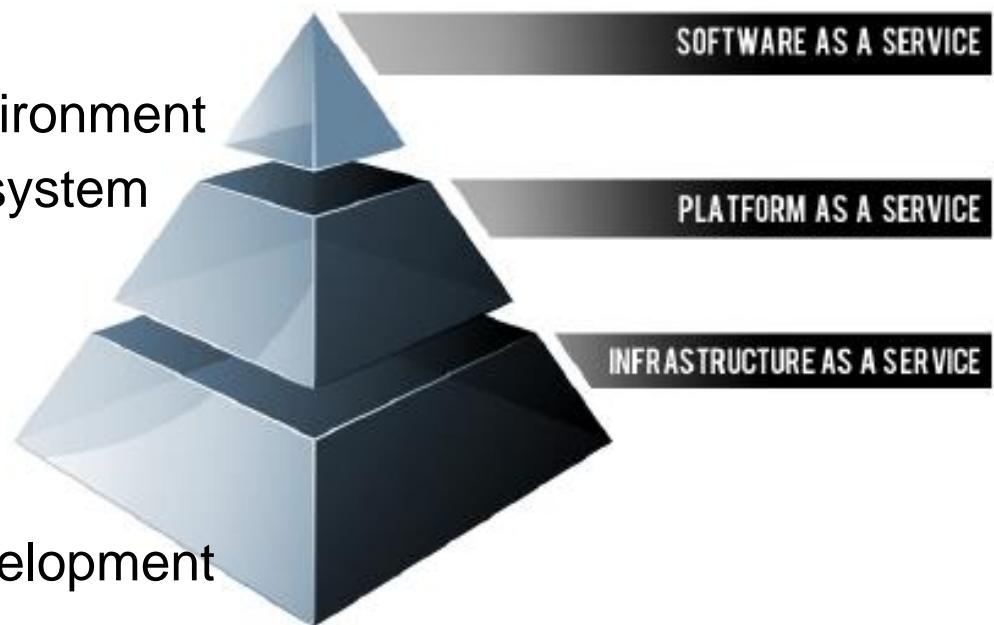
- Platform as a Service, referred to as PaaS, is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet.
- Platform as a Service allows users to create software applications using tools supplied by the provider.
- PaaS services are hosted in the cloud and accessed by users simply via their web browser.
- PaaS services can consist of preconfigured features that customers can subscribe to; they can choose to include the features that meet their requirements while discarding those that do not.

# Building blocks of PaaS

---

- PaaS providers can assist developers from the conception of their original ideas to the creation of applications, and through to testing and deployment.
- Below are some of the features that can be included with a PaaS offering:

- Operating system
- Server-side scripting environment
- Database management system
- Server Software
- Support
- Storage
- Network access
- Tools for design and development
- Hosting



# Characteristics of PaaS

---

- Services to develop, test, deploy, host and maintain applications in the same integrated development environment. All the varying services needed to fulfill the application development process
- Web based user interface creation tools help to create, modify, test and deploy different UI scenarios
- Multi-tenant architecture where multiple concurrent users utilize the same development application
- Built in scalability of deployed software including load balancing and failover
- Integration with web services and databases via common standards
- Support for development team collaboration – some PaaS solutions include project planning and communication tools
- Tools to handle billing and subscription management

# Characteristics of PaaS

---

PaaS, which is similar in many ways to Infrastructure as a Service, is differentiated from IaaS by the addition of value added services and comes in two distinct flavours;

1. A collaborative platform for software development, focused on workflow management regardless of the data source being used for the application. An example of this approach would be Heroku, a PaaS that utilizes the Ruby on Rails development language.
2. A platform that allows for the creation of software utilizing proprietary data from an application. This sort of PaaS can be seen as a method to create applications with a common data form or type. An example of this sort of platform would be the Force.com. PaaS from Salesforce.com which is used almost exclusively to develop applications that work with the Salesforce.com CRM

# Advantages and Risks

---

## Advantages

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

# Advantages and Risks

---

## Risks

- Since users rely on a provider's infrastructure and software, vendor lock-in can be an issue in PaaS environments.
- Other risks associated with PaaS are provider downtime or a provider changing its development roadmap.
- If a provider stops supporting a certain programming language, users may be forced to change their programming language, or the provider itself. Both are difficult and disruptive steps.

# Amazon Relational Database Service (RDS)

- Amazon Relational Database Service a web service that provides the capabilities of MySQL, Oracle, or Microsoft SQL Server relational database as a managed, cloud-based service
- On-demand provisioning
- No operating system for you to access
- Platforms:
  - MySQL
  - Oracle
  - SQL Server
  - PostgreSQL
- “Mostly” pre-configured
- Basic monitoring / metrics provided
- Automated backups
- Automated recovery
- User initiated “snapshots”
- Automated replication
- Software upgrades provided

# RDS Features and Functionality

- Pre-configured Parameters
- Monitoring and Metrics
- Automatic Software Patching
- Automated Backups
- DB Snapshots
- Push-Button Scaling
- Automatic Host Replacement
- Replication: two features
- Multi-AZ Deployment, Read Replica

# Oracle Deployment options

## RDS

- Quick provisioning
- “Easy” management
- Simple environment setup
- Simplified replication strategy
- No OS-level control
- Limited granular fine tuning
- Limited platform / versions

## EC2+Database

- You manage it yourself
- OS & storage overhead
- Software / version management
- Configuration
- Usually more costly
- More control over config / performance
- Allows more complex setups

# RDS Limitation

- Failovers are NOT instant.
  - Can take up to 6 minutes. Maybe more, depending on database size
- Only limited platforms / versions are supported.
- Upgrades / patching requires downtime.
  - You set an allowable maintenance window. Amazon will upgrade sometime during that window.
- Can suffer from “noisy neighbor” syndrome.
- Performance issues are sometimes hard to pinpoint
- Master-Master replication is NOT supported



**BITS** Pilani  
Pilani Campus

# BITS Pilani presentation

Mridul Moitra  
Cloud Computing

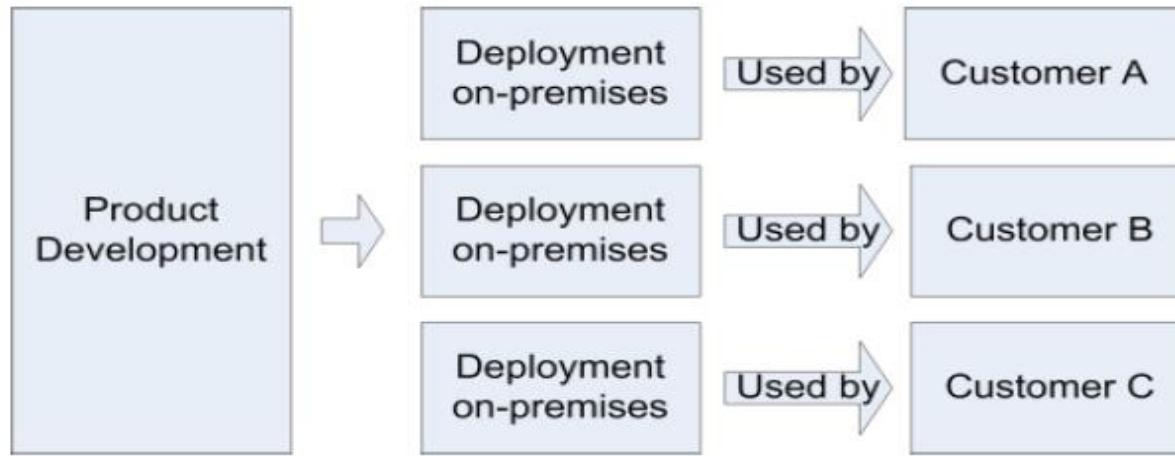




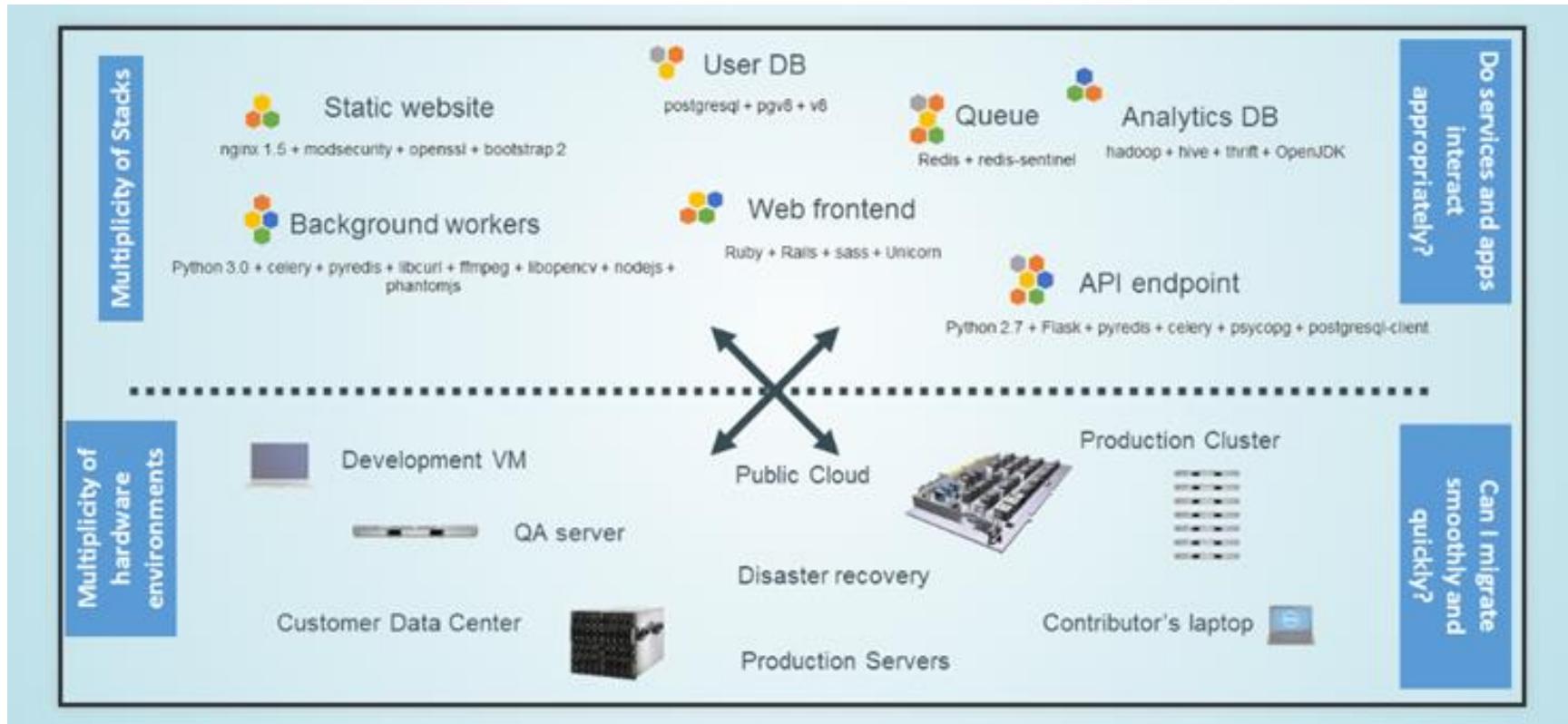
<CSI ZG527 / SS ZG527 / SE ZG527  
Cloud Computing- Docker Technology  
Lecture No. 5

**BITS** Pilani

# Traditional Deployment Model



# Challenges



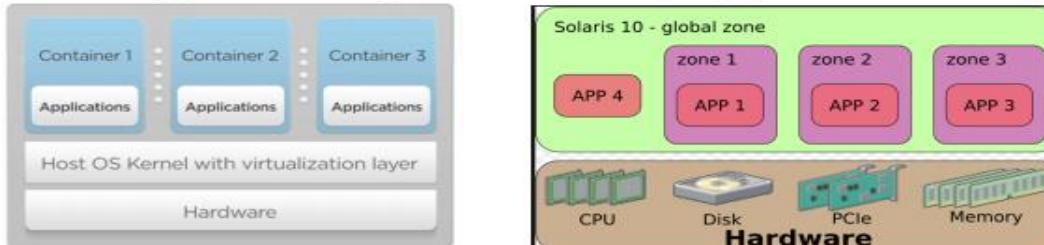
# Introduction

- Linux containers (LXC) are “lightweight” VMs
- Docker is a commoditized LXC technique that dramatically simplifies the use of LXC

# OS Virtualization

## OS Virtualization

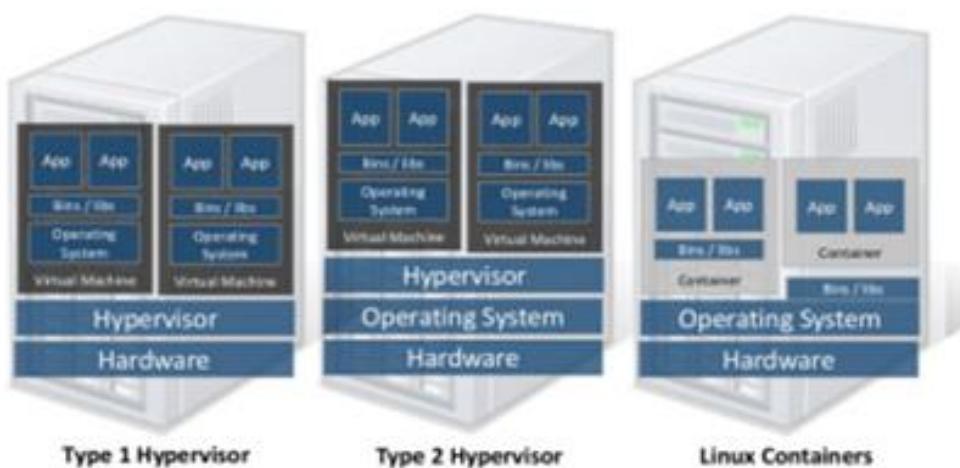
- Emulate OS-level interface with native interface
- “Lightweight” virtual machines
  - No hypervisor, OS provides necessary support



- Referred to as *containers*
  - Solaris containers, BSD jails, Linux containers

# Linux Container

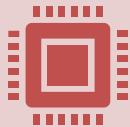
- Containers share OS kernel of the host
  - OS provides resource isolation
- Benefits
  - Fast provisioning, bare-metal like performance, lightweight



# OS Mechanisms for LXC

- OS mechanisms for resource isolation and management
- namespaces: process-based resource isolation
- Cgroups: limits, prioritization, accounting, control
- chroot: apparent root directory
- Linux security module, access control
- Tools (e.g., docker) for easy management

# Linux Namespaces



Linux kernel provides  
the “control groups”  
(cgroups) functionality

allows limitation and prioritization of  
resources (CPU, memory, block I/O,  
network, etc.) without the need for  
starting any VM



“namespace  
isolation” functionality

allows complete isolation of an  
applications' view of the operating  
environment,  
including process trees, networking, user  
IDs and mounted file systems

# Container Features

- Containers running in the user space
- Each container has
  - Own process space
  - Own network interface
  - Own /sbin/init (coordinates the rest of the boot process and configures the environment for the user)
  - Run stuff as root
- Share kernel with the host
- No device emulation

# Isolation with cgroups

- Memory
- Cpu
- Blkio
- devices

# Memory cgroup

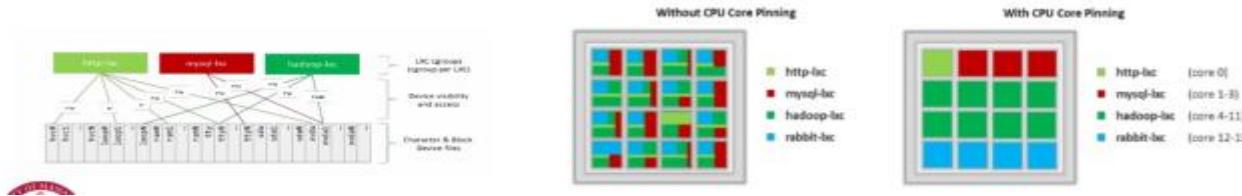
- keeps track pages used by each group:
  - file (read/write/mmap from block devices; swap)
  - anonymous (stack, heap, anonymous mmap)
  - active (recently accessed)
  - inactive (candidate for eviction)
- each page is charged to a group
- pages can be shared
- Individual (per-cgroup) limits and out-of-memory killer

## CPU cgroup

- keep track of user/system CPU time
- set relative weight per group
- pin groups to specific CPU(s)
  - Can be used to reserve CPUs for some apps

# Linux CGROUPS

- Resource isolation
  - what and how much can a container use?
    - Set upper bounds (limits) on resources that can be used
    - Fair sharing of certain resources
- Examples:
  - cpu: weighted proportional share of CPU for a group
  - cpuset: cores that a group can access
  - block io: weighted proportional block IO access
  - memory: max memory limit for a group



## Blkio cgroup

- keep track IOs for each block device
  - read vs write; sync vs async
- set relative weights
- set throttle (limits) for each block device
  - read vs write; bytes/sec vs operations/sec

## Devices cgroup

- controls read/write/mknod permissions
- typically:
  - allow: /dev/{tty,zero,random,null}...
  - deny: everything else
  - maybe: /dev/net/tun, /dev/fuse, /dev/kvm, /dev/dri...
- fine-grained control for GPU, virtualization, etc

## Almost no overhead

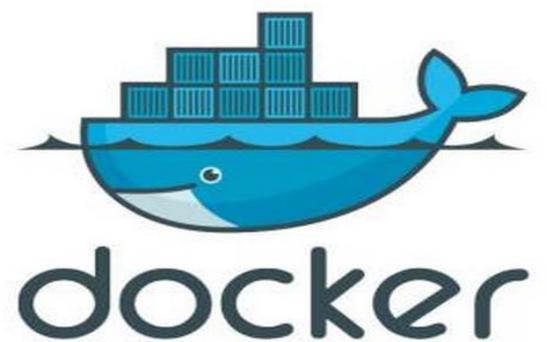
- Processes are isolated, but run straight on the host
- CPU performance = native performance
- Memory performance = a few % shaved off for (optional) accounting
- Network performance = small overhead; can be reduced to zero

# Proportional Share Scheduling

- Uses a variant of *proportional-share scheduling*
- *Share-based* scheduling:
  - Assign each process a weight  $w_i$  (a “share”)
  - Allocation is in proportional to share
  - fairness: reused unused cycles to others in proportion to weight
  - Examples: fair queuing, start time fair queuing
- *Hard limits*: assign upper bounds (e.g., 30%), no reallocation
- Credit-based: allocate credits every time  $T$ , can accumulate credits, and can burst up-to credit limit
  - can a process starve other processes?

# Docker

## Introduction and Demo



# Agenda

---

01

What is Docker

02

Why use Docker

03

How to setup Docker in Linux

04

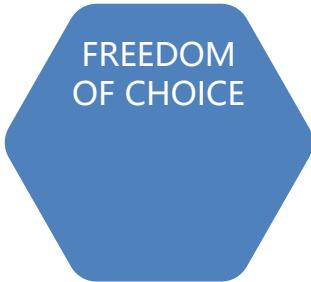
Commands & References

01

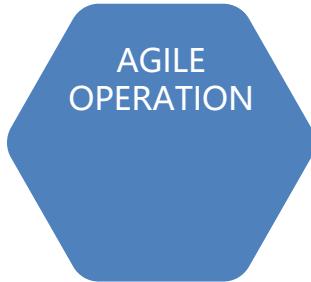
# What is Docker

## What is Docker - Overview

Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud. Today's businesses are under pressure to digitally transform but are constrained by existing applications and infrastructure while rationalizing an increasingly diverse portfolio of clouds, datacenters and application architectures. Docker enables true independence between applications and infrastructure and developers and IT ops to unlock their potential and creates a model for better collaboration and innovation.



FREEDOM  
OF CHOICE



AGILE  
OPERATION



INTEGRATED  
SECURITY

# Docker history

- 2013-03: Releases as Open Source
- 2013-09: Red Hat collaboration (Fedora, RHEL, OpenShift)
- 2014-03: 34th most starred GitHub project
- 2014-05: JAX Innovation Award (most innovative open technology)

# What is Docker?

Docker is a software platform that allows you to build, test, and deploy applications quickly, packaging software into standardized units called containers.

- Open Source engine to commoditize LXC
- using copy-on-write for quick provisioning
- allowing to **create and share** *images*
- **standard format** for containers
- standard, *reproducible* way to *easily* build *trusted* images  
(Dockerfile, Stackbrew...)

# What is Docker – Basic Concepts

## LXC

LXC(Linux Containers) is an operating-system-level virtualization method for running multiple isolated Linux systems(containers) on a control host using a single Linux kernel. [Wikipedia](#)

## CGroups & Namespaces

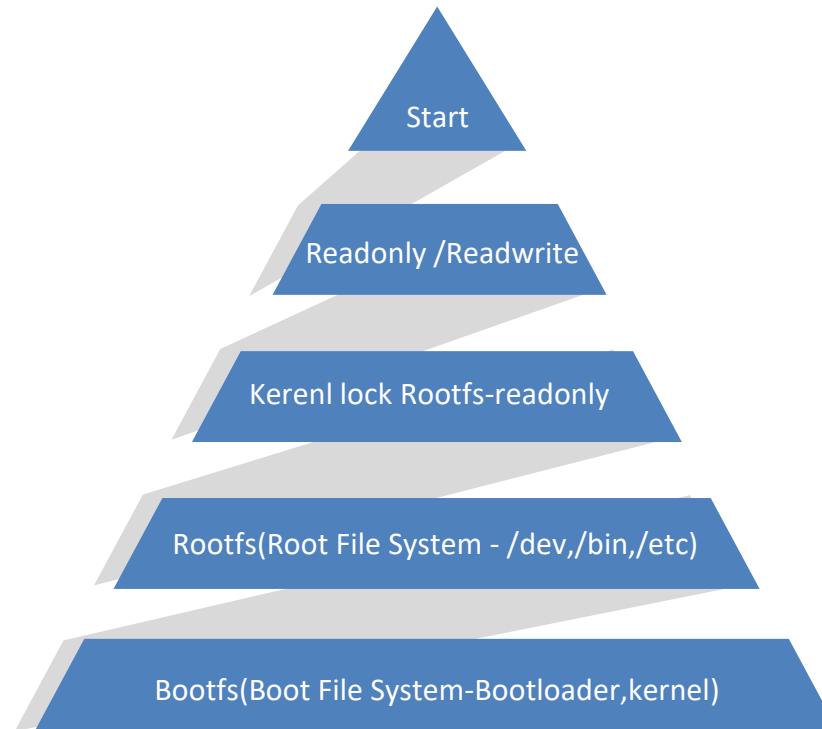
The [Linux kernel](#) provides the [cgroups](#) functionality that allows limitation and prioritization of resources (CPU, memory, block I/O, network, etc.) without the need for starting any [virtual machines](#), and also [namespace isolation](#) functionality that allows complete isolation of an applications' view of the operating environment, including [process](#) trees, [networking](#), [user IDs](#) and [mounted file systems](#)

## AUFS

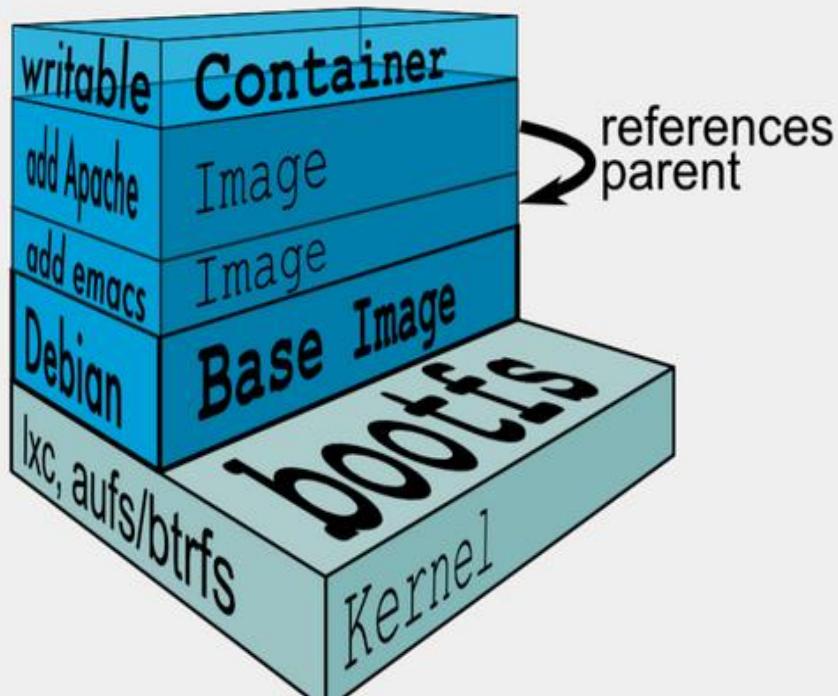
AUFS (short for advanced multi-layered unification filesystem) implements a [union mount](#) for [Linux file systems](#)

# What is Docker – Basic Concepts

## Linux Boot

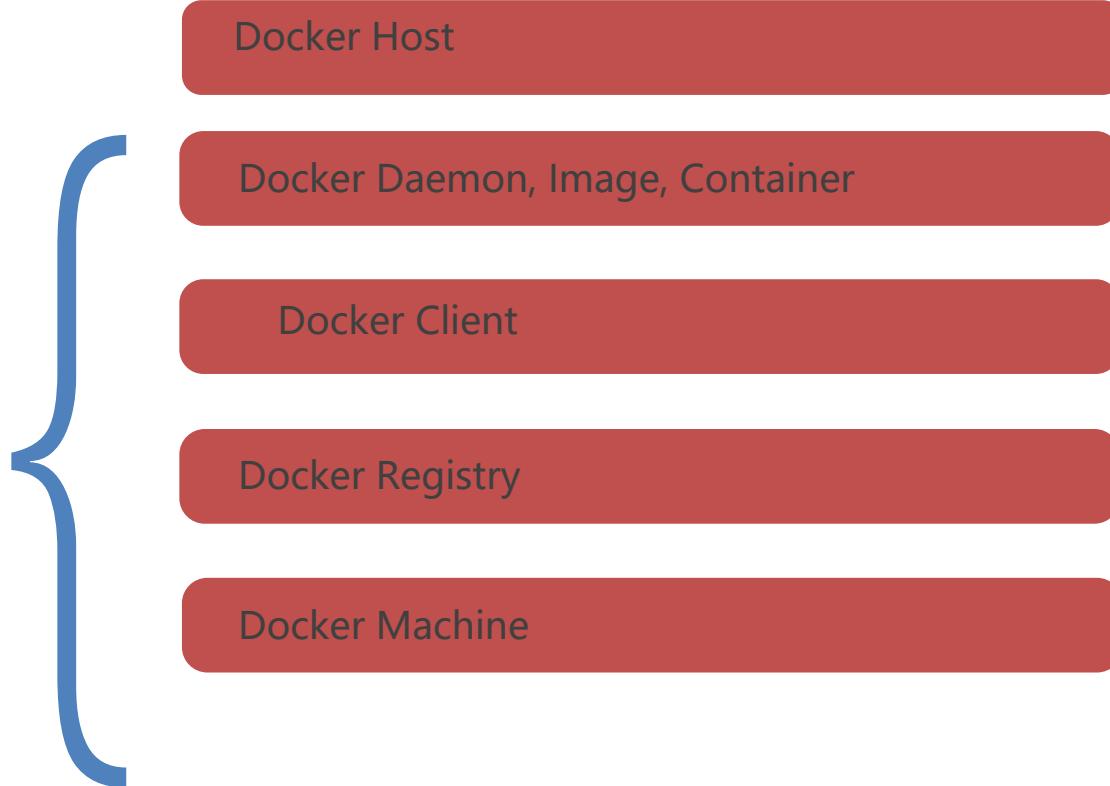
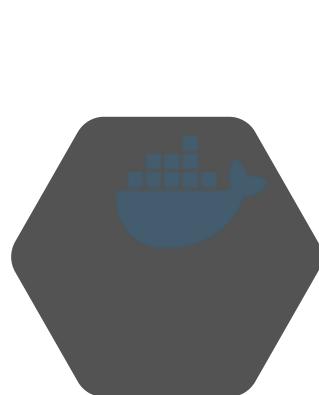


# Docker Images and Uses

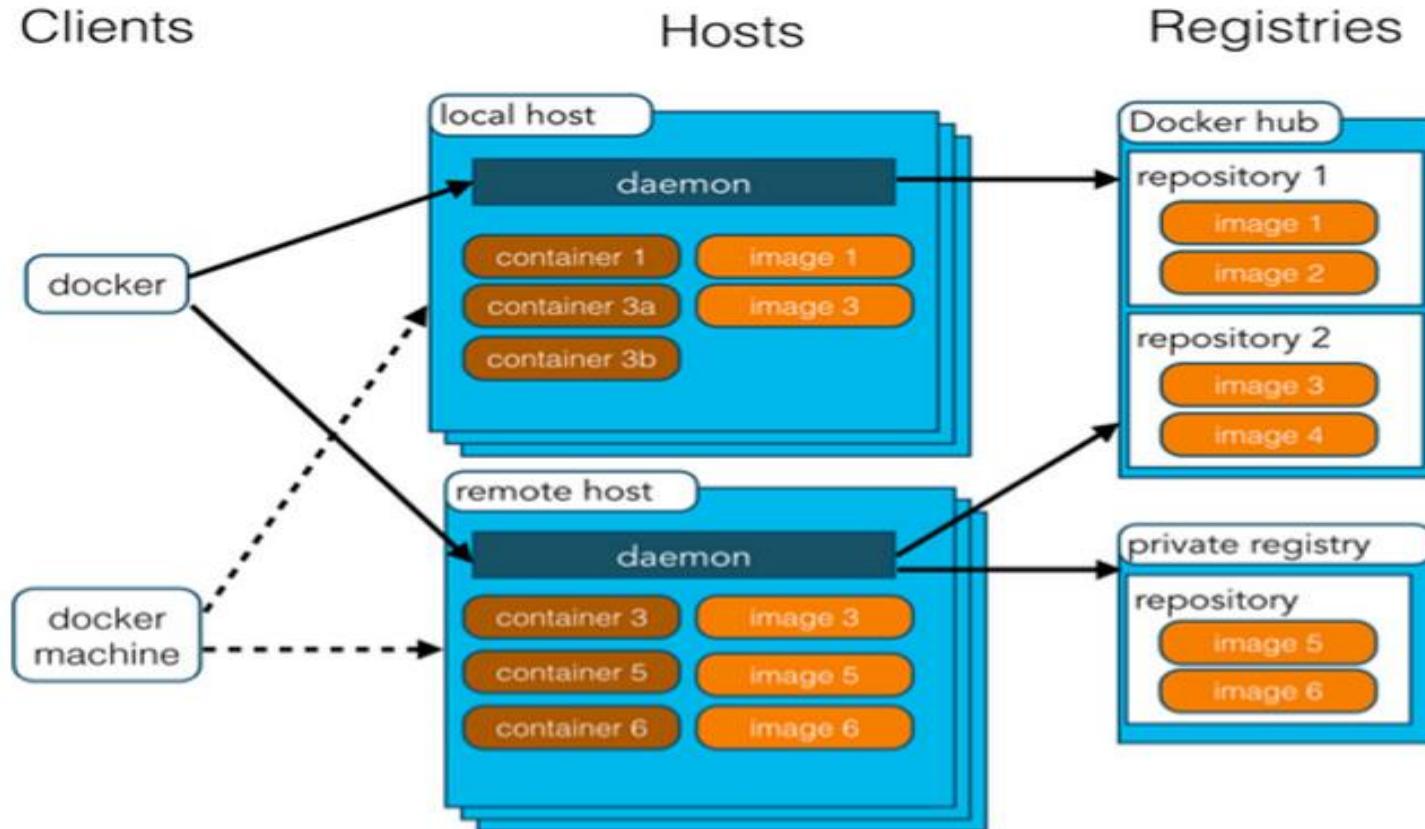


- Docker uses a union file system (AuFS)
  - allows containers to use host FS safely
- Essentially a copy-on-write file system
  - read-only files shared (e.g., share glibc)
  - make a copy upon write
- Allows for small efficient container images
- Docker Use Cases
  - “Run once, deploy anywhere”
  - Images can be pulled/pushed to repository
  - Containers can be a single process (useful for microservices) or a full OS

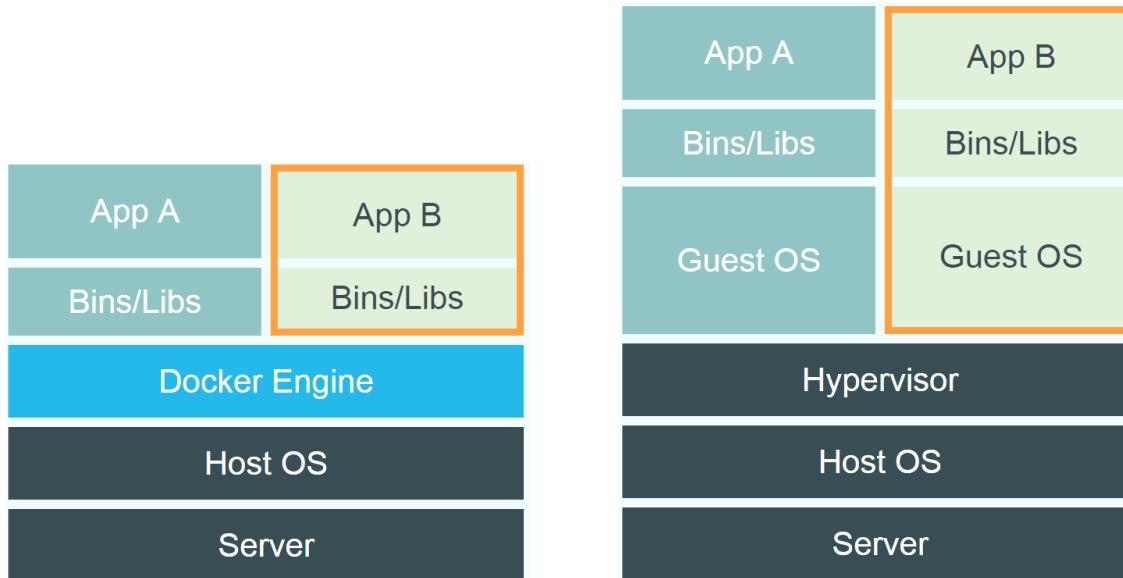
## What is Docker - Contains



## What is Docker - Contains



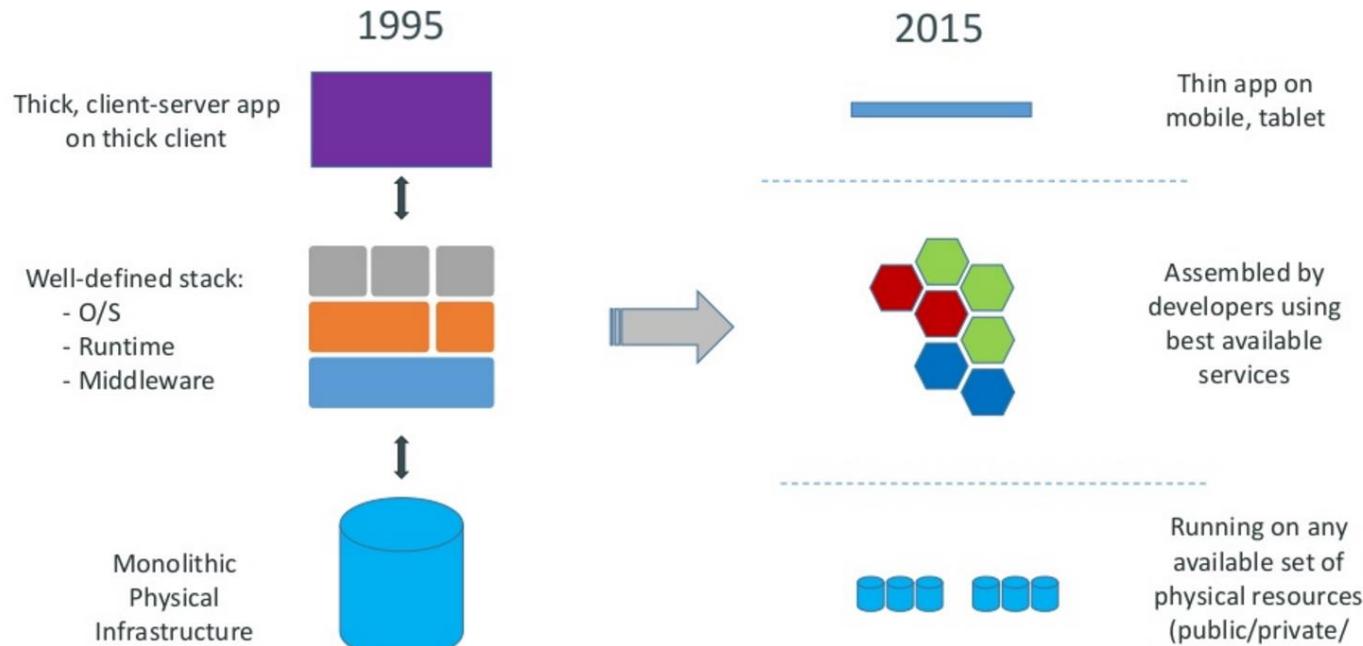
# Comparison between LXC/docker and VM



02

Why use Docker

# Motivation

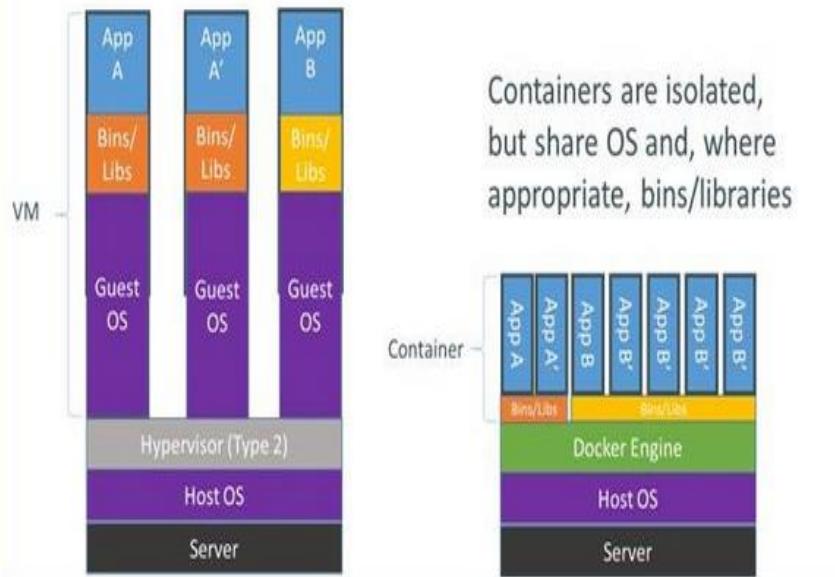


# Docker

- Minimal learning curve
- Rebuilds are easy
- Caching system makes rebuilds faster
- Single file to define the whole environment!

## Why use Docker – Compare with VM

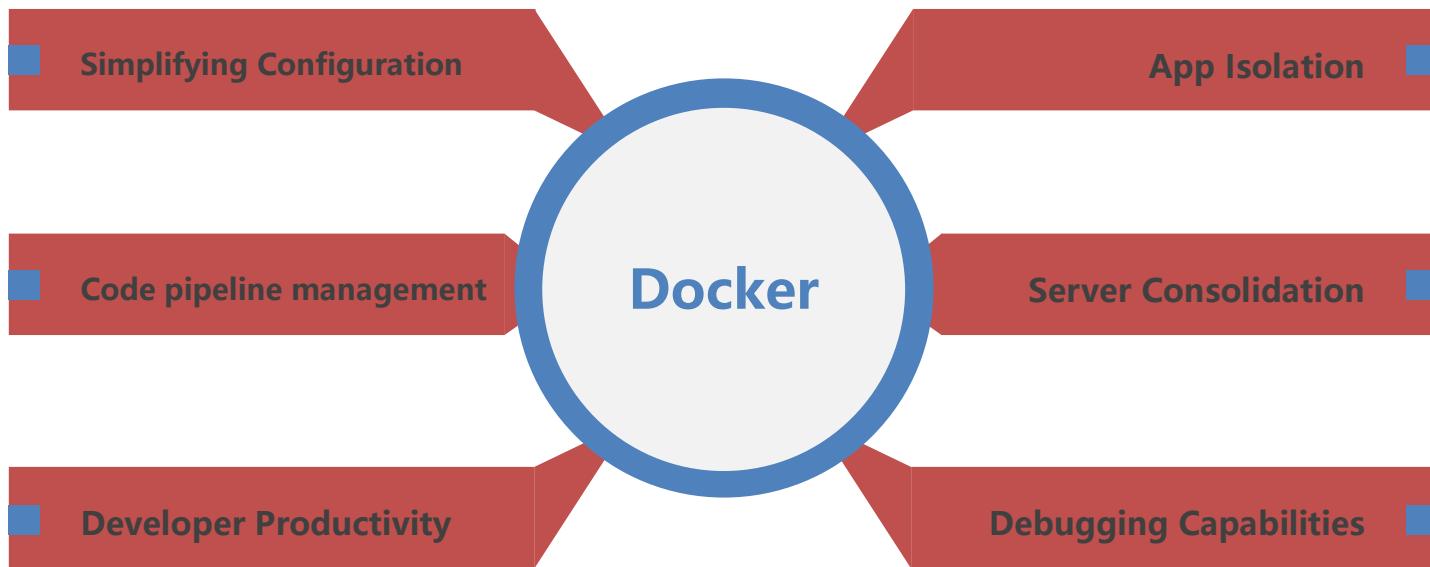
### Containers vs. VMs



Hyper-V (OS  
atleast 5G)  
VMWare, AWS EC2

Docker

## Why use Docker – Advantages



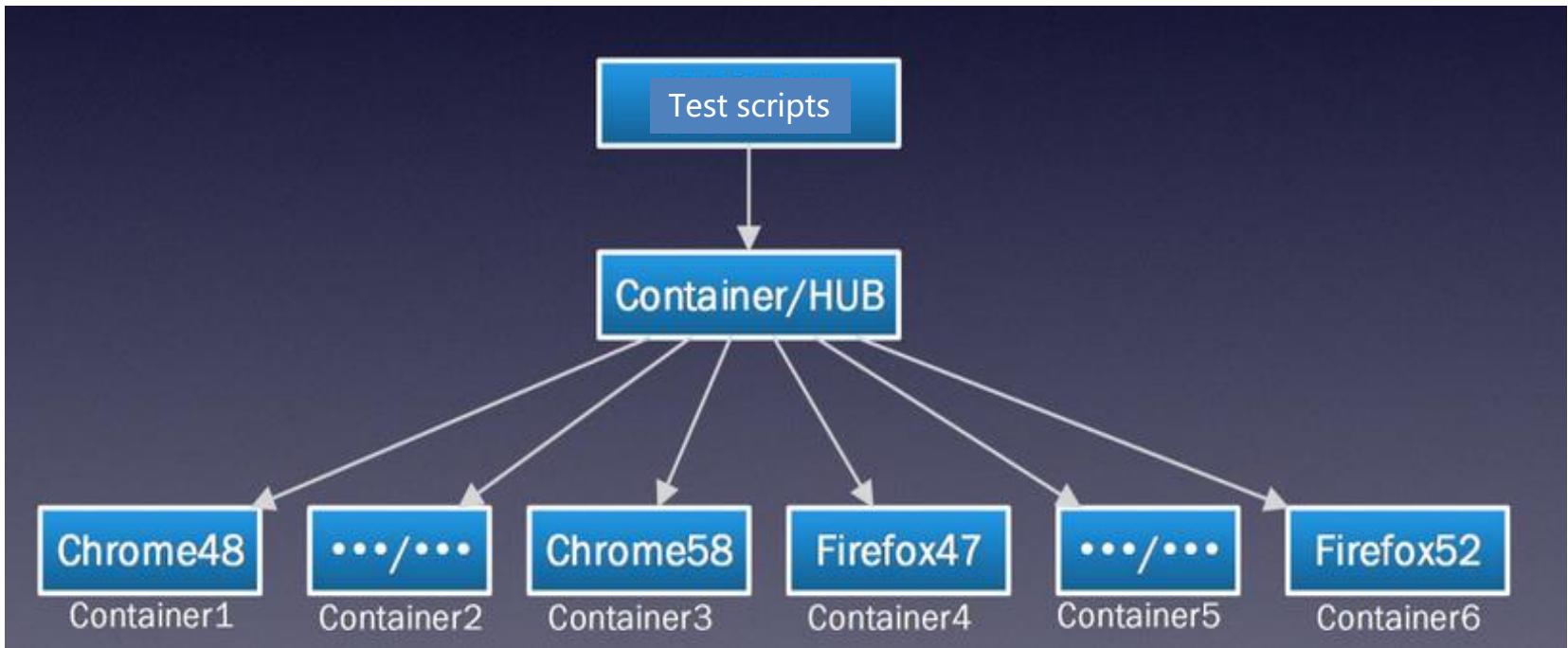
## Deploy Reliability and Consistently

- If it works locally, it will work on the server
- *With exactly the same behavior*
- Regardless of versions
- Regardless of distros
- Regardless of dependencies

## Deploy Efficiently

- Containers are lightweight
  - Typical laptop runs 10-100 containers easily
  - Typical server can run 100-1000 containers
- Containers can run at native speeds
  - Lies, damn lies, and other benchmarks:

## Why use Docker – Docker in Test



# Docker container—developer viewpoint

## Build once...run anywhere

- A clean, safe, hygienic and portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
- Automate testing, integration, packaging...anything you can script
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
- Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That's the power of Docker

# Administrative Benefits

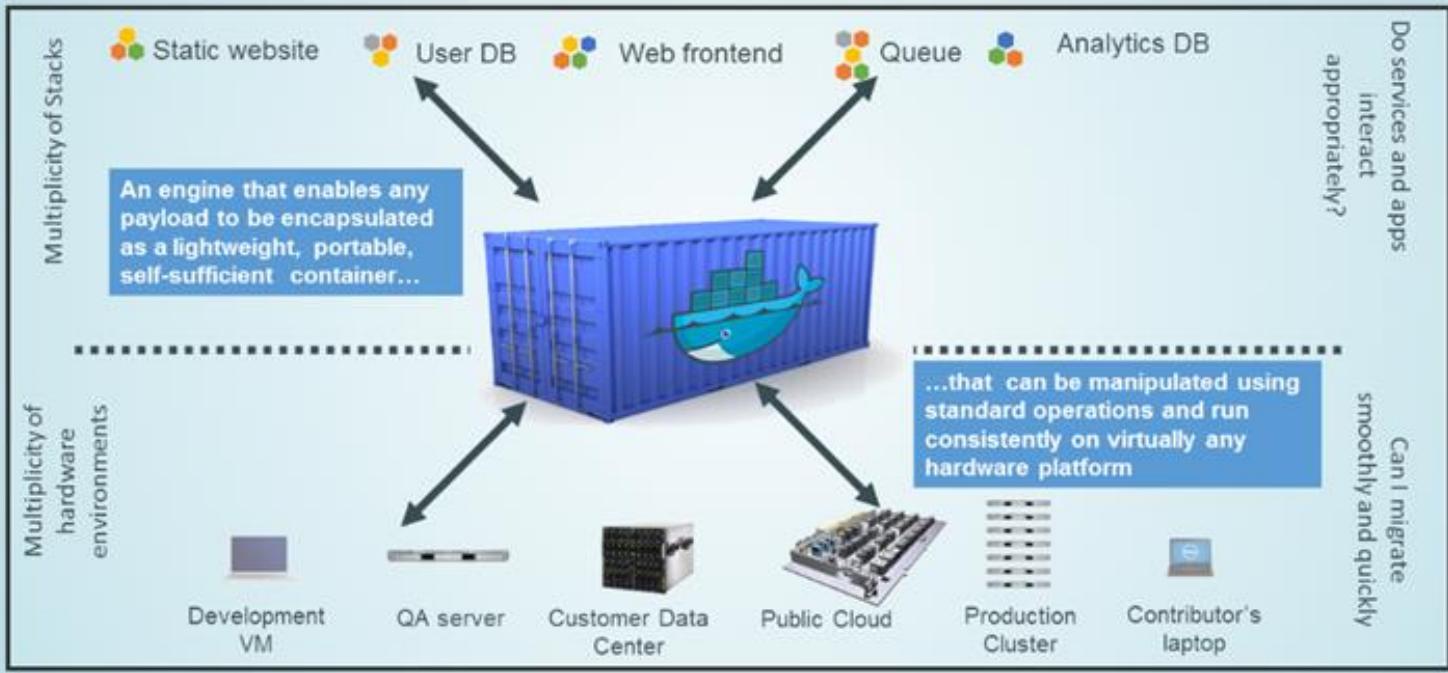
## Why Administrators Care

Configure once... run anything

- Make the entire lifecycle more efficient, consistent, and repeatable
- Increase the quality of code produced by developers.
- Eliminate inconsistencies between development, test, production, and customer environments.
- Support segregation of duties.
- Significantly improves the speed and reliability of continuous deployment and continuous integration systems.
- Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs.

# Docker Code Deployment

## Docker is a Container System for Code



# Docker Technical Details

## More Technical Details

### Why

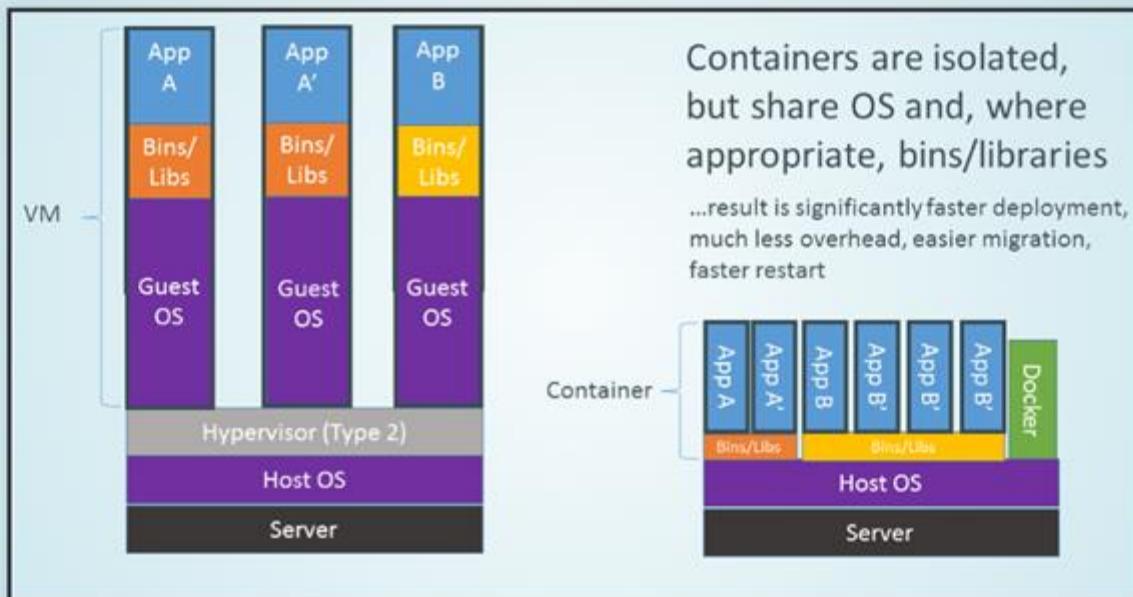
- Run everywhere
  - Regardless of kernel version
  - Regardless of host distro
  - Physical or virtual, cloud or not
  - Container and host architecture must match...
- Run anything
  - If it can run on the host, it can run in the container
  - If it can on a Linux kernel, it can run

### What

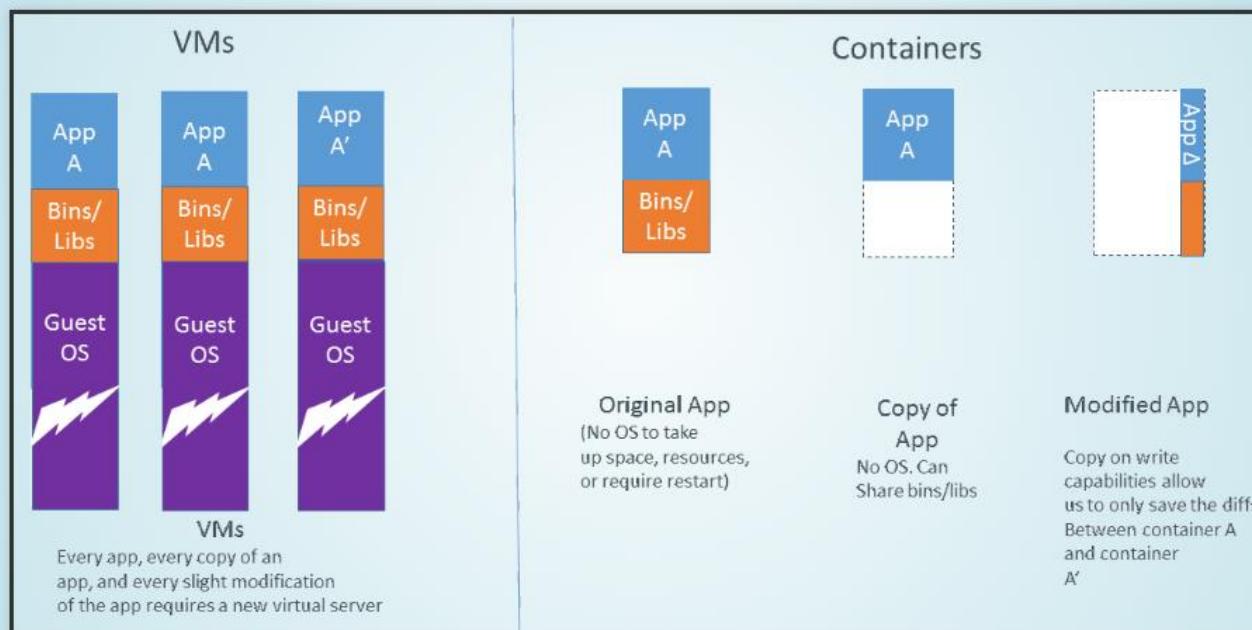
- High level: a lightweight VM
  - Own process space
  - Own network interface
  - Can run stuff as root
  - Can have its own /sbin/init (different from host)
  - <>machine container<>
- Low level: chroot on steroids
  - Can also *not* have its own /sbin/init
  - Container = isolated processes
  - Share kernel with host
  - <>application container<>

# Comparison between VMs vs Containers

## VMs vs Containers

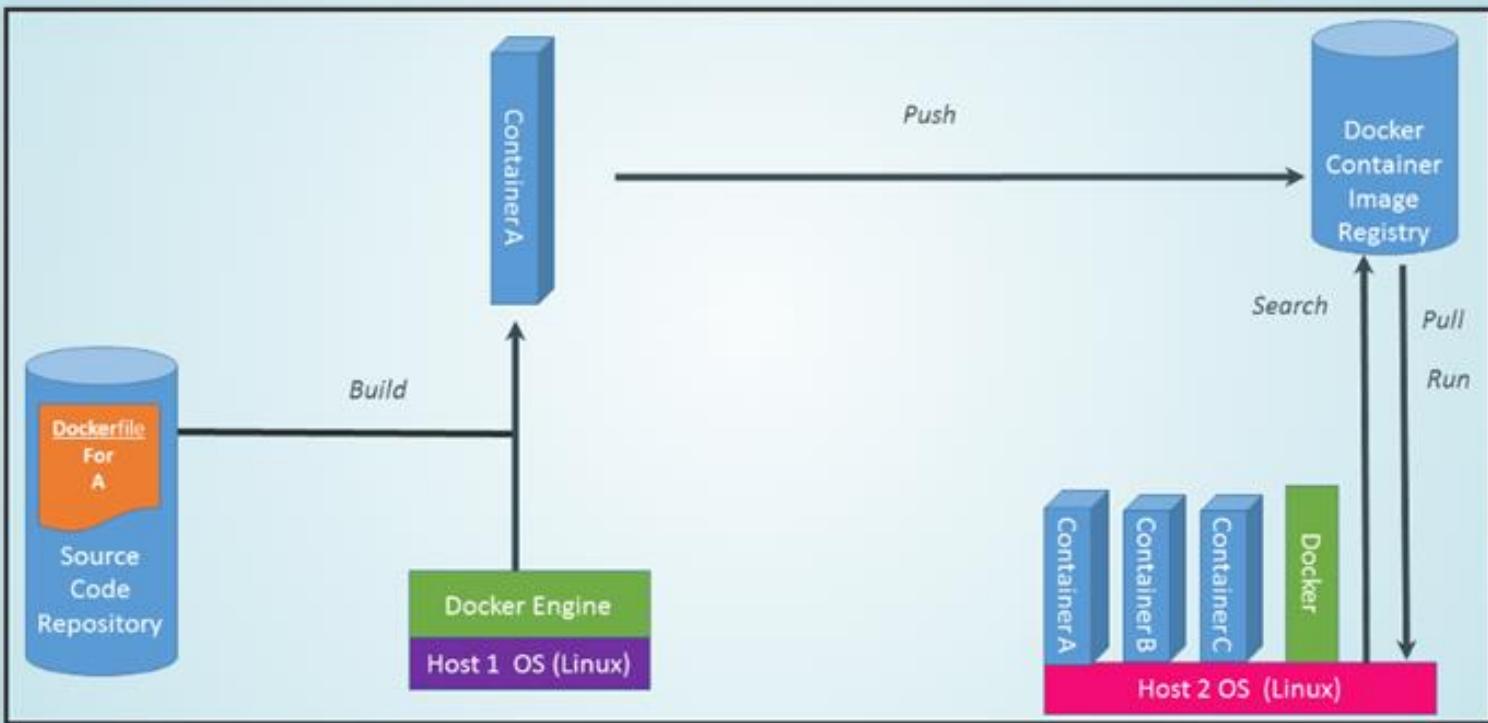


# Why are Docker Containers Lightweight?



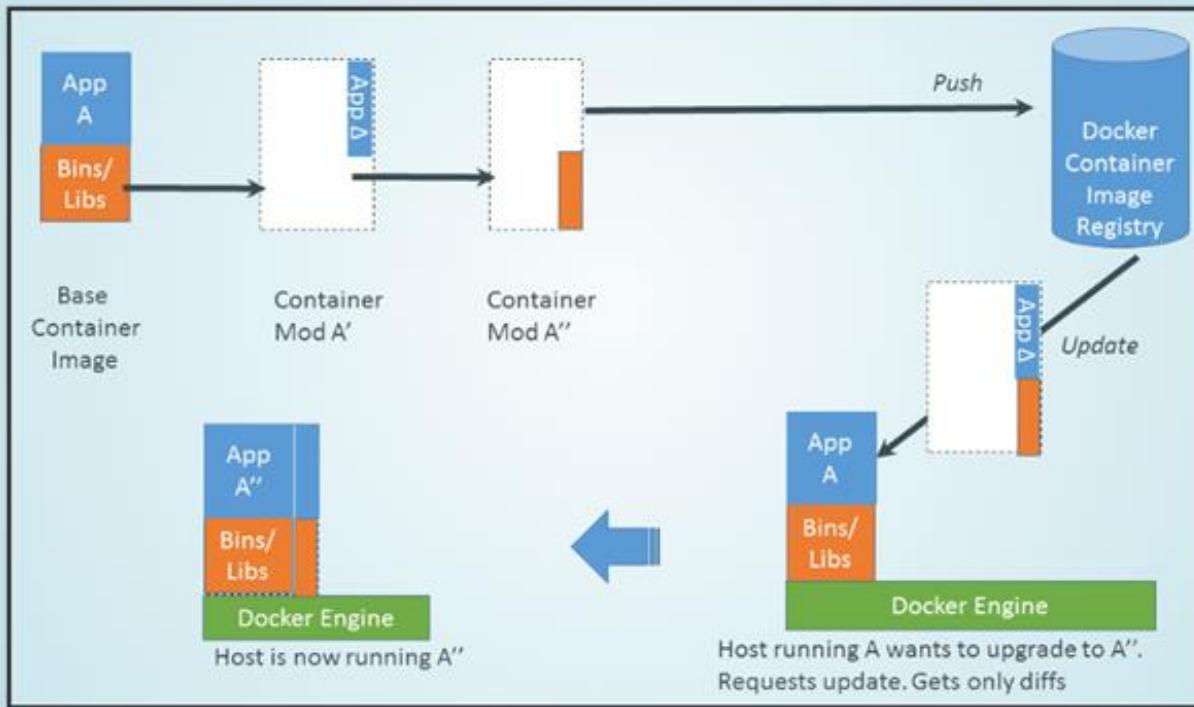
# Docker Deployment

## What are the Basics of a Docker System?



# Docker Changes

## Changes and Updates

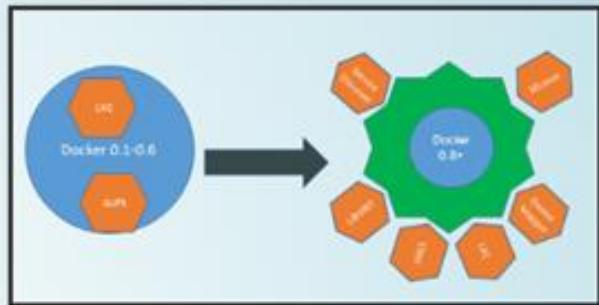


# Ecosystem Supports

- Operating systems
  - Virtually any distribution with a 2.6.32+ kernel
  - Red Hat/Docker collaboration to make work across RHEL 6.4+, Fedora, and other members of the family (2.6.32 +)
  - CoreOS—Small core OS purpose built with Docker
- OpenStack
  - Docker integration into NOVA (& compatibility with Glance, Horizon, etc.) accepted for Havana release
- Private PaaS
  - OpenShift, Solum (Rackspace, OpenStack), Other TBA
- Public PaaS
  - Deis, Voxoz, Cocaine (Yandex), Baidu PaaS
- Public IaaS
  - Native support in Rackspace, Digital Ocean,+++
  - AMI (or equivalent) available for AWS & other
- DevOps Tools
  - Integrations with Chef, Puppet, Jenkins, Travis, Salt, Ansible +++
- Orchestration tools
  - Mesos, Heat, ++
  - Shipyard & others purpose built for Docker
- Applications
  - 1000's of Dockerized applications available at index.docker.io

# Docker Futures

- Docker 0.7 (current release)
  - Fedora compatibility
  - Reduce kernel dependencies
  - Device mapper
  - Container linking
- Docker 0.8 (Dec)
  - Shrink and stabilize Core
  - Provide stable, pluggable API
  - RHEL compatibility
  - Nested containers
  - Beam: Introspection API based on Redis
  - Expand snapshot management features for data volumes
  - Will consider this “production ready”
- Docker 0.9 (Jan)
- Docker 1.0 (Feb)
  - Will offer support for this product



# Dockerfile

It is possible to build your own images reading instructions from a Dockerfile

```
FROM centos:7
RUN yum install -y python-devel python-virtualenv
RUN virtualenv /opt/indico/venv
RUN pip install indico
COPY entrypoint.sh /opt/indico/entrypoint.sh
EXPOSE 8000
ENTRYPOINT /opt/indico/entrypoint.sh
```

# docker-compose

Allows to run multi-container Docker applications reading instructions from a `docker-compose.yml` file

```
version: "2"
services:
  my-application:
    build: ./myapp
    ports:
      - "8000:8000"
    environment:
      - CONFIG_FILE
  db:
    image: postgres
  redis:
    image: redis
    command: redis-server --save "" --appendonly no
    ports:
      - "6379"
```

# Docker use cases

- Development Environment
- Environments for Integration Tests
- Quick evaluation of software
- Microservices
- Multi-Tenancy
- Unified execution environment (dev ↗ test ↗ prod (local, VM, cloud, ...))

# 03

## How to setup Docker in Linux

# How to setup Docker in Linux – Oracle linux 7.x

1. Touch file /etc/yum.repos.d/docker.repo

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/oraclelinux/7

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

2. Yum install -y docker-engine

3. Configure Firewall :

systemctl disable firewalld

yum install -y iptables-services

systemctl enable iptables

systemctl start iptables

4. Automatic start: systemctl enable docker.service

5. Manually start: systemctl start docker.service

6. Check status: systemctl status docker.service

# How to setup Docker in Linux – RHEL 7.4 build

```
[root@sgli-ddncaut03a ~]# uname -r  
3.10.0-693.5.2.el7.x86_64  
[root@sgli-ddncaut03a ~]# yum install docker  
Loaded plugins: product-id, search-disabled-repos
```

```
Installed:  
  docker.x86_64 2:1.12.6-61.git85d7426.el7  
  
Dependency Installed:  
  atomic-registries.x86_64 1:1.19.1-5.git48c224b.el7  
  audit-libs-python.x86_64 0:2.7.6-3.el7  
  checkpolicy.x86_64 0:2.5-4.el7  
  container-selinux.noarch 2:2.28-1.git85ce147.el7  
  container-storage-setup.noarch 0:0.7.0-1.git4ca59c5.el7  
  docker-client.x86_64 2:1.12.6-61.git85d7426.el7  
  docker-common.x86_64 2:1.12.6-61.git85d7426.el7  
  docker-rhel-push-plugin.x86_64 2:1.12.6-61.git85d7426.el7  
  json-glib.x86_64 0:1.2.6-1.el7  
  libcgROUP.x86_64 0:0.41-13.el7  
  libsemanage-python.x86_64 0:2.5-8.el7  
  oci-register-machine.x86_64 1:0-3.13.gitcd1e331.el7  
  oci-systemd-hook.x86_64 1:0.1.14-1.git1ba44c6.el7  
  oci-umount.x86_64 2:2.0.0-1.git299e781.el7  
  policycoreutils-python.x86_64 0:2.5-17.1.el7  
  python-IPy.noarch 0:0.75-6.el7  
  setools-libs.x86_64 0:3.3.8-1.1.el7  
  skopeo-containers.x86_64 1:0.1.24-1.dev.git28d4e08.el7  
  subscription-manager-plugin-container.x86_64 0:1.19.23-1.el7_4  
  yajl.x86_64 0:2.0.4-4.el7  
  
Complete!  
[root@sgli-ddncaut03a ~]#
```

# How to setup Docker in Linux – RHEL 7.4 build

```
[root@sg1i-ddncaut03a ~]# docker version
Client:
Version:      1.12.6
API version:  1.24
Package version: docker-1.12.6-61.git85d7426.el7.x86_64
Go version:   go1.8.3
Git commit:   85d7426/1.12.6
Built:        Tue Sep 26 15:30:51 2017
OS/Arch:      linux/amd64

Server:
Version:      1.12.6
API version:  1.24
Package version: docker-1.12.6-61.git85d7426.el7.x86_64
Go version:   go1.8.3
Git commit:   85d7426/1.12.6
Built:        Tue Sep 26 15:30:51 2017
OS/Arch:      linux/amd64
[root@sg1i-ddncaut03a ~]# █
```

# How to setup Docker in Linux – RHEL 7.4 build

```
[root@sgli-ddncaut03a ~]# docker search oracle
INDEX      NAME                           DESCRIPTION                                              STARS   OFFICIAL   AUTOMATED
docker.io   docker.io/oraclelinux          Official Docker builds of Oracle Linux.                448     [OK]
docker.io   docker.io/frolvlad/alpine-oraclejdk8  The smallest Docker image with OracleJDK 8...        303     [OK]
docker.io   docker.io/sath89/oracle-12c    Oracle Standard Edition 12c Release 1 with...       291     [OK]
docker.io   docker.io/alexeiled/docker-oracle-xe-11g This is a working (hopefully) Oracle XE 11...       251     [OK]
docker.io   docker.io/sath89/oracle-xe-11g  Oracle xe 11g with database files mount su...       180     [OK]
docker.io   docker.io/jasperon/oracle-11g  Docker image for Oracle 11g database                 63      [OK]
docker.io   docker.io/isuper/java-oracle   This repository contains all java releases...       55      [OK]
docker.io   docker.io/vmameless/oracle-xe-11g Dockerfile of Oracle Database Express Edit...       51      [OK]
docker.io   docker.io/oracle/glassfish     GlassFish Java EE Application Server on Or...       42      [OK]
docker.io   docker.io/oracle/openjdk      Docker images containing OpenJDK Oracle Linux       37      [OK]
docker.io   docker.io/airdock/oracle-jdk  Docker Image for Oracle Java SDK (8 and 7)...       30      [OK]
docker.io   docker.io/ingensi/oracle-jdk  Official Oracle JDK installed on centos.           21      [OK]
docker.io   docker.io/cogniteev/oracle-java Oracle JDK 6, 7, 8, and 9 based on Ubuntu ...       20      [OK]
docker.io   docker.io/n3ziniuk5/ubuntu-oracle-jdk Ubuntu with Oracle JDK. Check tags for ver...       16      [OK]
docker.io   docker.io/oracle/nosql        Oracle NoSQL on a Docker Image with Oracle...       16      [OK]
docker.io   docker.io/sgrrio/java-oracle  Docker Images of Java 7/8/9 provided by Or...       11      [OK]
docker.io   docker.io/andreptb/oracle-java Debian Jessie based image with Oracle JDK ...       7       [OK]
docker.io   docker.io/openweb/oracle-tomcat A fork off of Official tomcat image with 0...       7       [OK]
docker.io   docker.io/flurdy/oracle-java7 Base image containing Oracle's Java 7 JDK            5       [OK]
docker.io   docker.io/martinseeler/oracle-server-jre Oracle's Java 8 as 61 MB Docker container.       4       [OK]
docker.io   docker.io/davidcaste/debian-oracle-java Oracle Java 8 (and 7) over Debian Jessie         3       [OK]
docker.io   docker.io/teradatalabs/centos6-java8-oracle Docker image of CentOS 6 with Oracle JDK 8...       3       [OK]
docker.io   docker.io/spansari/nodejs-oracledb nodejs with oracledb installed globally on...       2       [OK]
docker.io   docker.io/publicisworldwide/oracle-core This is the core image based on Oracle Lin...       1       [OK]
docker.io   docker.io/softwareplant/oracle  oracle db                                         0       [OK]
[root@sgli-ddncaut03a ~]#
```

```
[root@sgli-ddncaut03a ~]# docker pull docker.io/oraclelinux
Using default tag: latest
Trying to pull repository docker.io/library/oraclelinux ...
latest: Pulling from docker.io/library/oraclelinux
1b19d6599a70: Downloading [=====] 82.89 MB/86.06 MB
```

```
[root@sgli-ddncaut03a ~]# docker pull docker.io/oraclelinux
Using default tag: latest
Trying to pull repository docker.io/library/oraclelinux ...
latest: Pulling from docker.io/library/oraclelinux
1b19d6599a70: Pull complete
Digest: sha256:6067eb9ac8edc2042508e2adfd00b9fb1cc1d38e708d0f5f98058a8740ba0661
[root@sgli-ddncaut03a ~]#
```



# How to setup Docker in Linux – RHEL 7.4 build

```
[root@sgli-ddncaut03a ~]# docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
docker.io/oraclelinux  latest  af8cf7fc5b7e  2 weeks ago  233.7 MB
[root@sgli-ddncaut03a ~]#
```

```
[root@sgli-ddncaut03a ~]# docker inspect af8cf7fc5b7e
[
  {
    "Id": "sha256:af8cf7fc5b7e9e4dcee6db022f23118d98ff54bfea1458bfb2d2ad7fad61770f",
    "RepoTags": [
      "docker.io/oraclelinux:latest"
    ],
    "RepoDigests": [
      "docker.io/oraclelinux@sha256:6067eb9ac8edc2042508e2adfd00b9fb1cc1d38e708d0f5f98058a8740ba0661"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2018-04-18T18:40:12.935217168Z",
    "Container": "4b74c7f189be3a3f2b318729e06b963df935520e5544520c2b26c17fddf63f55",
    "ContainerConfig": {
      "Hostname": "4b74c7f189be",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [
        "/bin/sh",
        "-c",
        "#(nop) ",
        "CMD [\"/bin/bash\"]"
      ]
    }
  }
]
```



# How to setup Docker in Linux – RHEL 7.4 build

```
[root@sgli-ddncaut03a ~]# docker run --help  
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]  
Run a command in a new container
```

```
[root@sgli-ddncaut03a ~]# docker run -i -t docker.io/oraclelinux /bin/bash  
[root@9524b56297f5 ~]# echo "hello docker"  
hello docker  
[root@9524b56297f5 ~]# ls  
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
```

```
[root@sgli-ddncaut03a ~]# docker ps  
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS          NAMES  
9524b56297f5        docker.io/oraclelinux   "/bin/bash"   About a minute ago   Up About a minute   zen_chandrasekhar  
[root@sgli-ddncaut03a ~]# docker ps -a  
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS          NAMES  
9524b56297f5        docker.io/oraclelinux   "/bin/bash"   About a minute ago   Up About a minute   zen_chandrasekhar  
[root@sgli-ddncaut03a ~]#
```

# Commands & Reference – Docker Commands



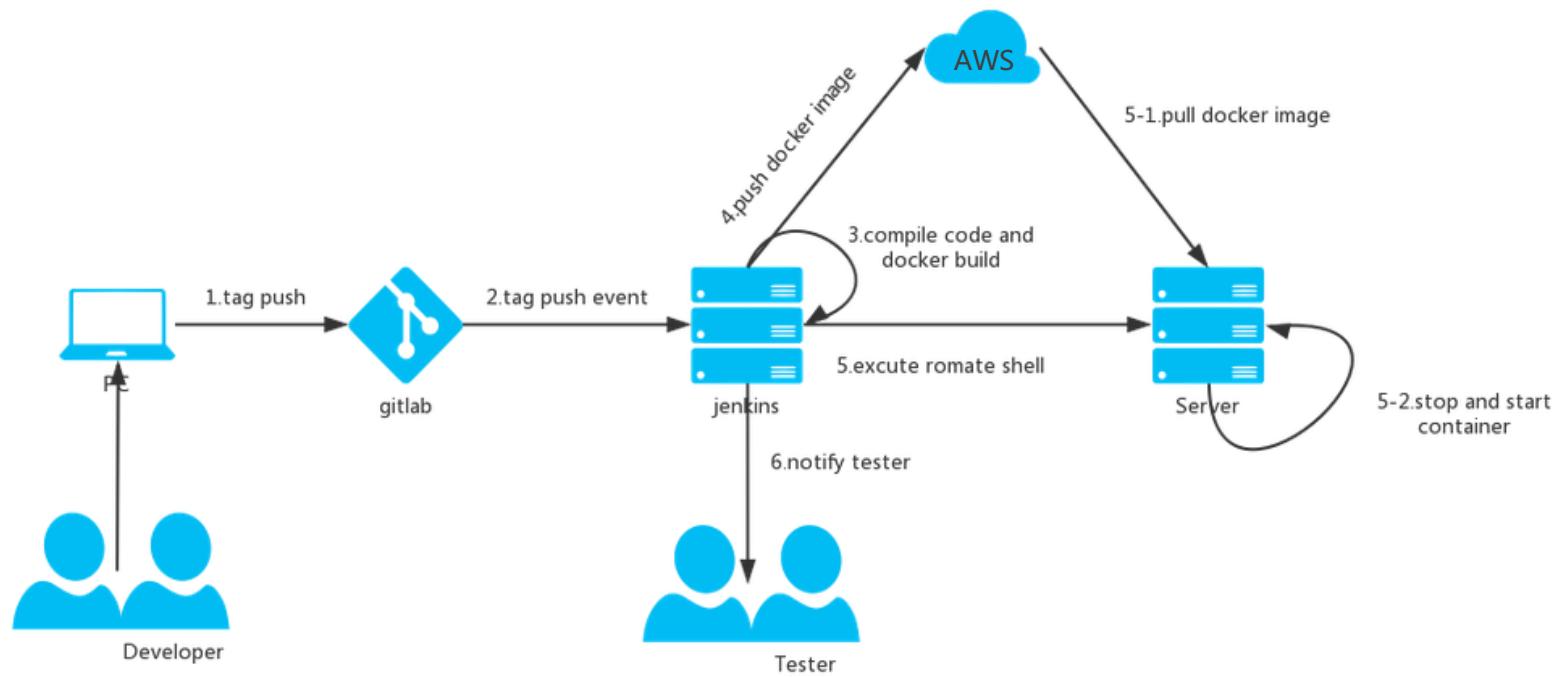
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]	runoob@runoob:~\$ docker run -it nginx:latest /bin/bash
docker start [OPTIONS] CONTAINER [CONTAINER...]	docker start myrunoob
docker stop [OPTIONS] CONTAINER [CONTAINER...]	docker stop myrunoob
docker restart [OPTIONS] CONTAINER [CONTAINER...]	docker restart myrunoob
docker kill [OPTIONS] CONTAINER [CONTAINER...]	runoob@runoob:~\$ docker kill -s KILL mynginx
docker rm [OPTIONS] CONTAINER [CONTAINER...]	docker rm -f db01 db02
docker create [OPTIONS] IMAGE [COMMAND] [ARG...]	runoob@runoob:~\$ docker create --name myrunoob nginx:latest
docker ps [OPTIONS]	runoob@runoob:~\$ docker ps -a -q
docker inspect [OPTIONS] NAME ID [NAME ID...]	runoob@runoob:~\$ docker inspect mysql:5.6
docker top [OPTIONS] CONTAINER [ps OPTIONS]	runoob@runoob:~\$ docker top mymysql
docker logs [OPTIONS] CONTAINER	runoob@runoob:~\$ docker logs -f mynginx
docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]	runoob@runoob:~\$ docker commit -a "runoob.com" -m "my apache" a404c6c174a2 mymysql:v1
docker diff [OPTIONS] CONTAINER	runoob@runoob:~\$ docker diff mymysql
docker login [OPTIONS] [SERVER]	docker login -u username -p password
docker pull [OPTIONS] NAME[:TAG]@[DIGEST]	docker pull java
docker push [OPTIONS] NAME[:TAG]	docker push myapache:v1
docker search [OPTIONS] TERM	runoob@runoob:~\$ docker search -s 10 java
docker images [OPTIONS] [REPOSITORY[:TAG]]	runoob@runoob:~\$ docker images
docker build [OPTIONS] PATH   URL   -	docker build -t runoob/ubuntu:v1 .

# Commands & Reference – Dockerfile Commands



FROM	FROM <image>:<tag>
MAINTAINER	MAINTAINER <name>
RUN	RUN <command> (the command is run in a shell - '/bin/sh -c')
CMD	CMD ["executable", "param1", "param2"] (like an exec, this is the preferred form)
ENTRYPOINT	ENTRYPOINT ["executable", "param1", "param2"] (like an exec, the preferred form)
USER	ENTRYPOINT ["memcached", "-u", "daemon"]
EXPOSE	EXPOSE <port> [<port>...]
ENV	ENV <key> <value>
ADD	ADD <src> <dest>
VOLUME	VOLUME ["<mountpoint>"]

# Why use Docker – Docker in Devops



## Commands & References – References



<http://www.docker.com>

<https://docs.docker.com/linux/>

<https://docs.docker.com/engine/userguide/>

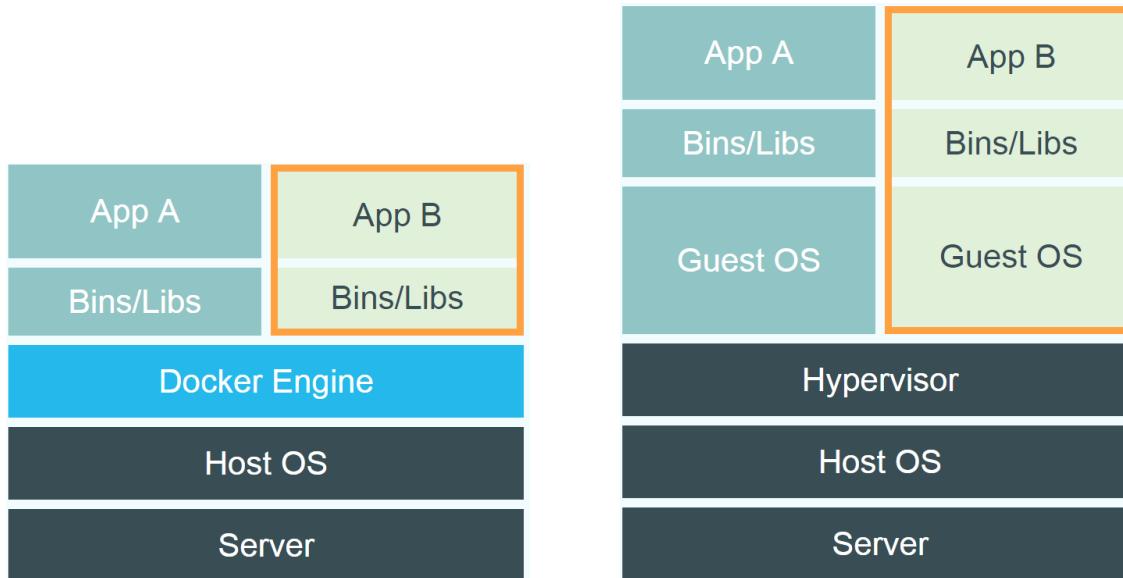
<https://www.docker.com/open-source>

<https://hub.docker.com/>

<https://www.docker-cn.com/>

<https://thehub.thomsonreuters.com/docs/DOC-2572951>

# Comparison between LXC/docker and VM



# Docker Engine

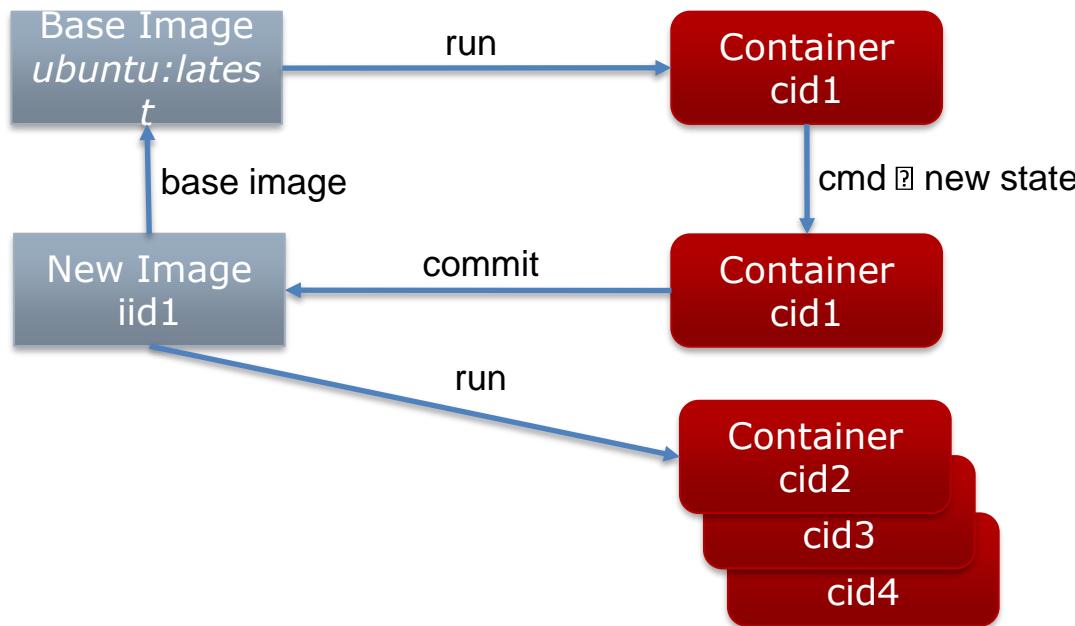
- the Docker engine runs in the background
  - manages containers, images, and builds
  - HTTP API (over UNIX or TCP socket)
  - embedded CLI talking to the API

# Building docker image

## □ With run/commit commands

- 1) docker run ubuntu bash
- 2) apt-get install this and that
- 3) docker commit <containerid> <imagename>
- 4) docker run <imagename> bash
- 5) git clone git://.../mycode
- 6) pip install -r requirements.txt
- 7) docker commit <containerid> <imagename>
- 8) repeat steps 4-7 as necessary
- 9) docker tag <imagename> <user/image>
- 10) docker push <user/image>

# Docker



# Authoring image with a dockerfile

## □ A sample dockerfile

**FROM ubuntu**

```
RUN apt-get -y update
RUN apt-get install -y g++
RUN apt-get install -y erlang-dev erlang-manpages erlang-base-
hipe ...
RUN apt-get install -y libmozjs185-dev libicu-dev libtool ...
RUN apt-get install -y make wget
RUN wget http://.../apache-couchdb-1.3.1.tar.gz | tar -C /tmp -
zxf-
RUN cd /tmp/apache-couchdb-* && ./configure && make install
RUN printf "[httpd]\nport = 8101\nbind_address = 0.0.0.0" >
/usr/local/etc/couchdb/local.d/docker.ini
```

**EXPOSE 8101**

**CMD ["/usr/local/bin/couchdb"]**

**Run the command to build:**

**docker build -t your\_account/couchdb .**

# Docker Hub

- Public repository of Docker images
  - <https://hub.docker.com/>
  - docker search [term]
- Automated: Has been automatically built from Dockerfile
  - Source for build is available on GitHub

## Dev-> test->production

- code in local environment  
(« dockerized » or not)
- each push to the git repo triggers a hook
- the hook tells a build server to clone the code and run « docker build » (using the Dockerfile)
- the containers are tested (nosetests, Jenkins...),  
and if the tests pass, pushed to the registry
- production servers pull the containers and run them
- for network services, load balancers are updated

# Docker has a repository like github

you can push and pull container images to/from  
the Docker registry

which is something like a “GitHub” for Docker  
container images.

# Docker has a repository like github

you can push and pull container images to/from  
the Docker registry

which is something like a “GitHub” for Docker  
container images.

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

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

## Multitenancy – key aspects

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

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

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

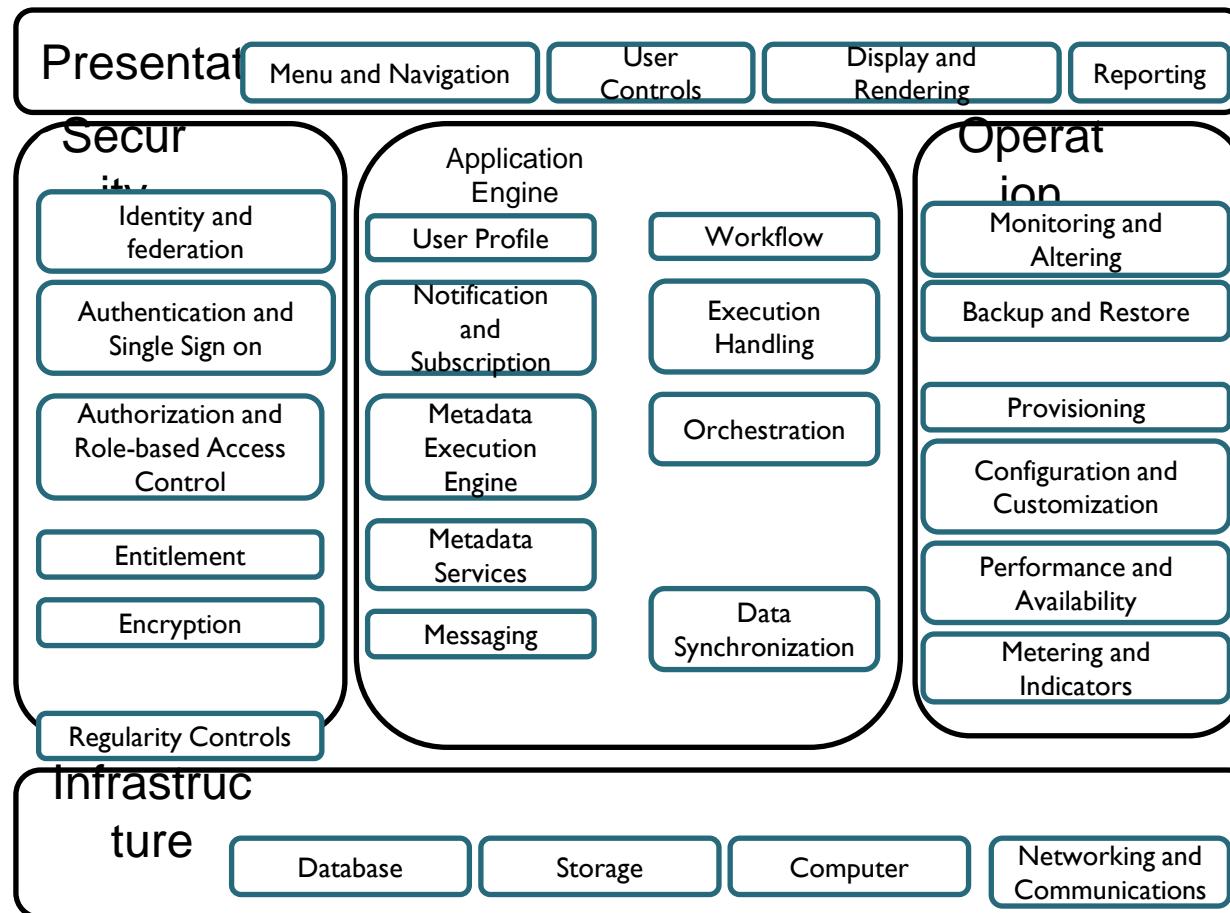
# Multi-tenants Deployment Modes for Application Server

<b>Fully isolated Application server</b> Each tenant accesses an application server running on a dedicated servers.	<p>The diagram shows two separate vertical stacks of circles. The top stack is labeled 'Application Server' and the bottom stack is also labeled 'Application Server'. The left stack is labeled 'Tenant A' and the right stack is labeled 'Tenant B'. Lines connect each tenant to its respective application server.</p>
<b>Virtualized Application Server</b> Each tenant accesses a dedicated application running on a separate virtual machine.	<p>The diagram shows two separate vertical stacks of circles. The top stack is labeled 'Application server' and the bottom stack is also labeled 'Application server'. The left stack is labeled 'Tenant A' and the right stack is labeled 'Tenant B'. Lines connect each tenant to its respective application server. The application servers are shown within boxes labeled 'Virtual machine'.</p>
<b>Shared Virtual Server</b> Each tenant accesses a dedicated application server running on a shared virtual machine.	<p>The diagram shows two separate vertical stacks of circles. The top stack is labeled 'Application server' and the bottom stack is also labeled 'Application server'. The left stack is labeled 'Tenant A' and the right stack is labeled 'Tenant B'. Lines connect each tenant to its respective application server. The application servers are shown within a single large orange box labeled 'Virtual machine'.</p>
<b>Shared Application Server</b> The tenant shared the application server and access application resources through separate session or threads.	<p>The diagram shows two separate vertical stacks of circles. The top stack is labeled 'Session thread' and the bottom stack is also labeled 'Session thread'. The left stack is labeled 'Tenant A' and the right stack is labeled 'Tenant B'. Lines connect each tenant to its respective session thread. The session threads are shown within a single large orange circle labeled 'Application Server' and 'Session Thread'.</p>

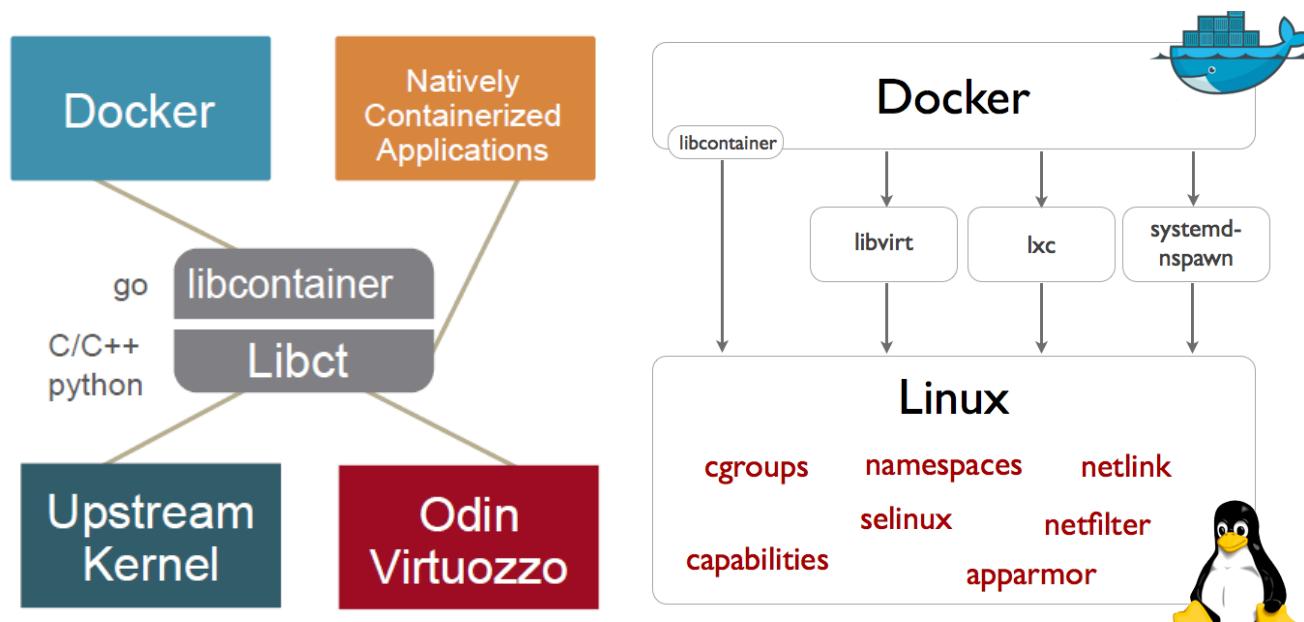
# Multi-tenants Deployment Modes in Data Centers

<p><b>Fully isolated data center</b> The tenants do not share any data center resources</p>	<p>Tenant A</p> <p>Tenant B</p>
<p><b>Virtualized servers</b> The tenants share the same host but access different databases running on separate virtual machines</p>	<p>Tenant A</p> <p>Tenant B</p> <p>Virtual Machine</p> <p>Database</p> <p>Virtual Machine</p> <p>Database</p>
<p><b>Shared Server</b> The tenants share the same server (Hostname or IP) but access different databases</p>	<p>Tenant A</p> <p>Tenant B</p>
<p><b>Shared Database</b> The tenants share the same server and database (shared or different ports) but access different schema (tables)</p>	<p>Tenant A</p> <p>Tenant B</p>
<p><b>Shared Schema</b> The tenants share the same server, database and schema (tables). The irrespective data is segregated by key and rows.</p>	<p>Tenant A</p> <p>Tenant B</p>

# Conceptual framework of Software as a Service



# Docker needs containers





# Cloud Computing

## SEWP ZG527

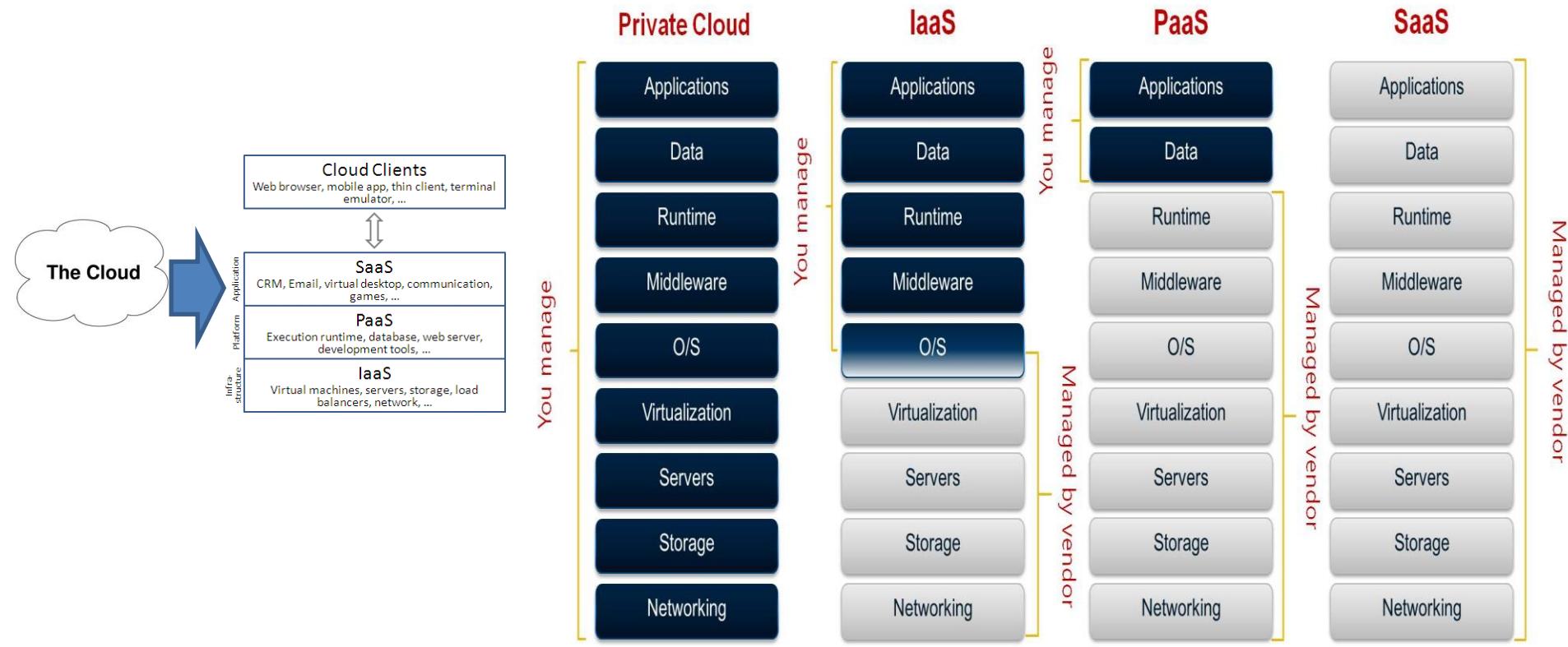
**BITS** Pilani

# Introduction

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



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



---

- Cloud **Distributed environment**

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

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





**OMG!!!!**



**Users/  
Brokers**

### SLA Resource Allocator

**Service Request Examiner and Admission Control**  

- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management

**Pricing**

**Accounting**

**VM Monitor**

**Dispatcher**

**Service Request Monitor**

### Virtual Machines (VMs)

### Physical Machines

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



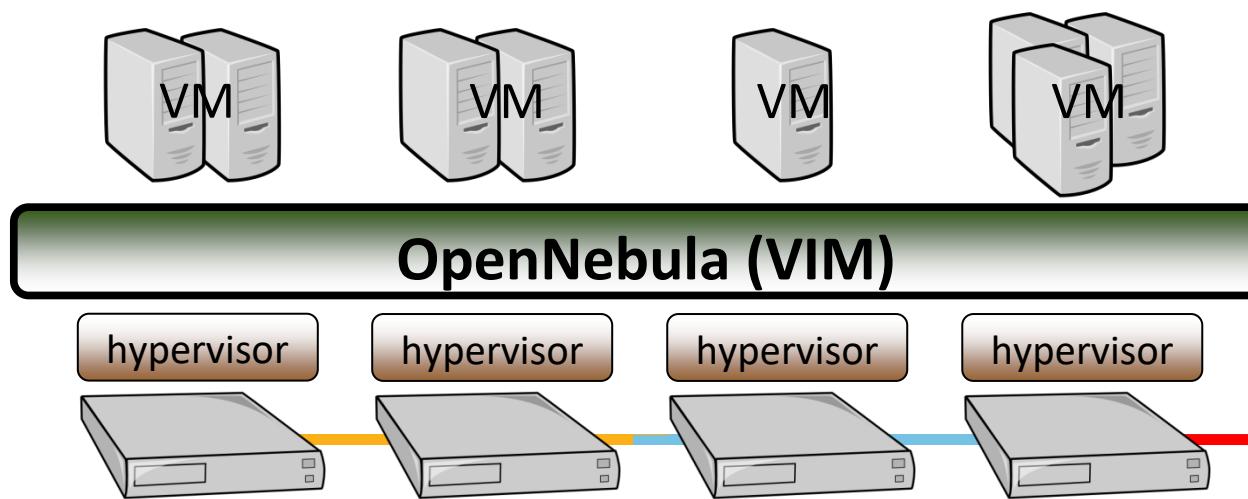


Need of the hour???

# Virtual Infrastructure Managers

# Why a Virtual Infrastructure Manager?

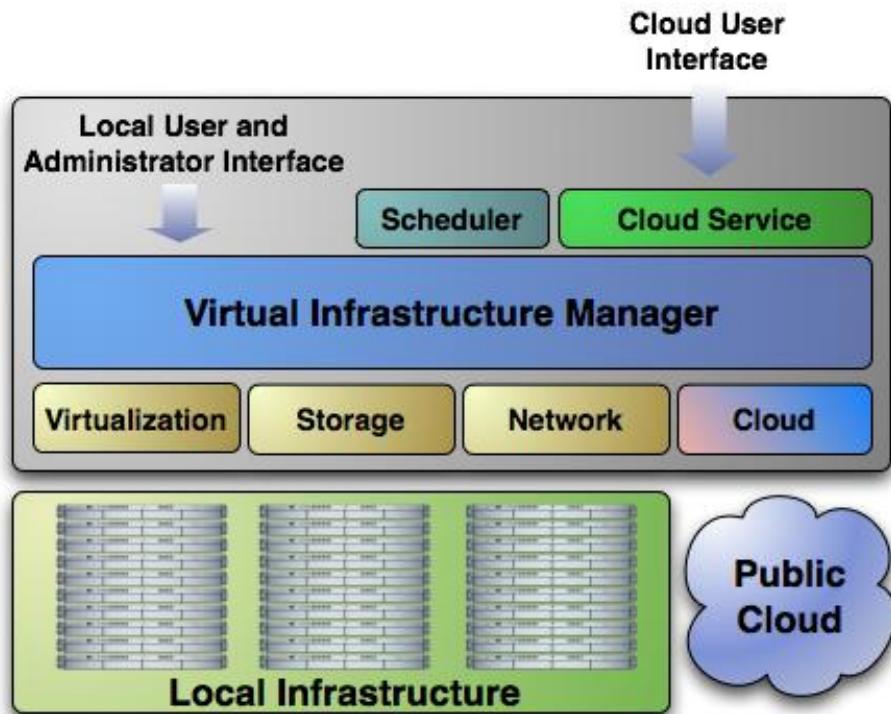
- VMs are great!!...but something more is needed
  - Where did/do I put my VM? (**scheduling & monitoring**)
  - How do I provision a new cluster node? (**clone**)
  - What IP addresses are available? (**networking**)
- Provide a **uniform view** of the resource pool
- **Life-cycle management** and monitoring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



# Extending the Benefits of Virtualization to Clusters

---

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

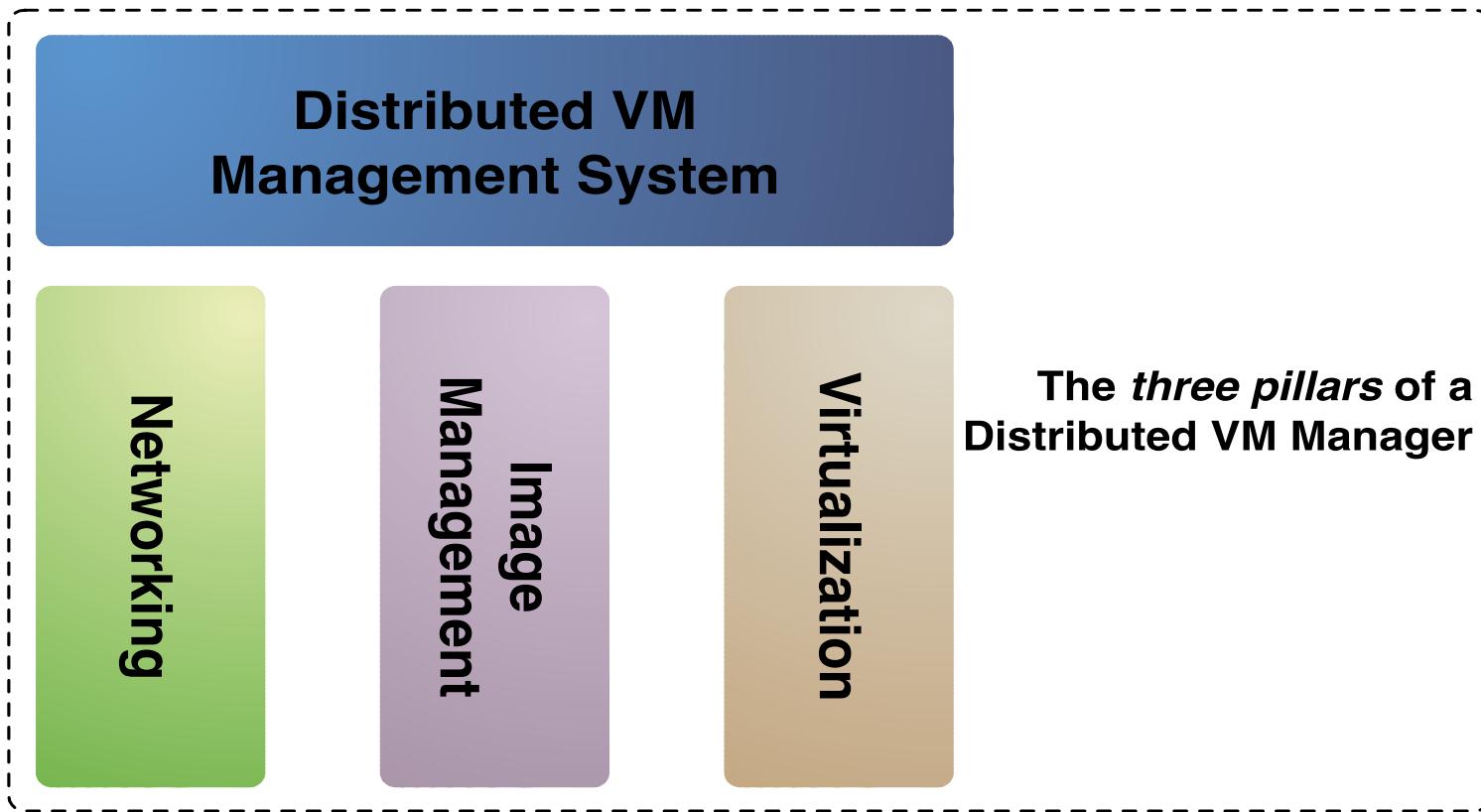


- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

# Virtual Machine Management Model

---

## Distributed VM Management Model



# What is OpenNebula?

## What is OpenNebula?

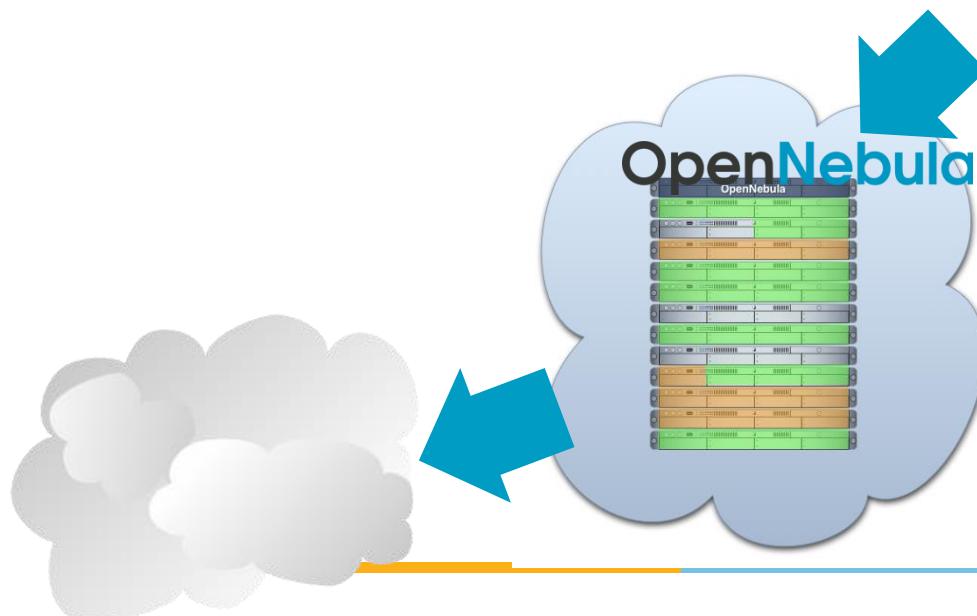
---

- OpenNebula is a cloud computing platform for managing heterogeneous distributed data center infrastructures. The OpenNebula platform manages a data center's virtual infrastructure to build private, public and hybrid implementations of Infrastructure as a Service
  - It's a powerful, but easy-to-use, **open source** platform for your enterprise **private**, **hybrid** or **edge** cloud infrastructure, you are at the right place.
  - **OpenNebula**, the Cloud & Edge Computing Platform that unifies public cloud simplicity and agility with private cloud performance, security and control. OpenNebula brings flexibility, scalability, simplicity, and vendor independence to support the growing needs of your developers and DevOps practices
-

# What is OpenNebula?

Enabling Technology to Build your Cloud

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



# The Open Source Cloud & Edge Computing Platform for the Enterprise



## Any Application

Automate operations and manage VMs & Kubernetes on a single shared environment



## Any Infrastructure

Open cloud architectures based on KVM, LXC system containers, or Firecracker microVMs



## Any Cloud

A powerful **multi-cloud** platform that combines **private, public, and edge** cloud operations

# Open Nebula Technology

---



## LIGHT & SIMPLE

Lightweight and easy to install, maintain, operate, upgrade and use



## FLEXIBLE

Fully open-source and customizable to fit into any data center and policies



## ROBUST

Production-ready, highly-scalable, reliable and supported



## POWERFUL

Innovative functionality for private/hybrid clouds and DC virtualization

- ✓ Only one OpenNebula distribution
  - ✓ Fully open-source, enterprise-ready
  - ✓ Delivered as a single, upgradable package
-

# Open Nebula Features

## Cloud Management

- VDC multi-tenancy
- Simple cloud GUI and interfaces
- Service elasticity/provisioning
- Federation/hybrid

OpenNebula

## Virtual Infra Management

- Capacity management
- Multi-VM management
- Resource optimization
- HA and business continuity

OpenNebula

vCenter

KVM

Xen

VMware



# Open Nebula Features

## THE CLOUD CONSUMER PERSPECTIVE



### Cloud View

- Simple one-click provisioning portal
- Pre-loaded set of configurations
- Access to usage information and quotas



### Flow

- Multi-VM applications
- Elasticity Rules
- Easy parameterization of VMs



### Showback

- Cost reports for resource usage
- Set by CPU, Memory or disk usage
- Associated with VM Templates



### Interfaces & APIs

- CLI tools
- Multiple API Bindings

# Open Nebula Features

## THE CLOUD ADMIN PERSPECTIVE



### Interfaces

- Sunstone GUI
- Powerful CLI



### Deployment and Integration

- Simple installation and upgrade
- Hook system and HA configurations
- VM contextualization



### Virtual Infrastructure Operation

- Life cycle management VMs
- Virtual Network and Storage Consumption and Delegation



### Capacity Management

- Scheduling policies
- User & Groups
- Accounting, quotas, permissions & ACLs

# Open Nebula Features

## THE CLOUD ARCHITECT PERSPECTIVE



### Federation of Multiple DCs

- User access to multiple DCs
- Local management of DC resources
- Minimal synch. to tolerate big latencies



### Hybrid Cloud Computing

- Seamless integration of external cloud
- Local & Remote VM profiles
- Allocation policies



### Provisioning of Virtual Datacenters

- Group users, resources and infrastructure
- Delegate management of virtual resources
- Partition/isolation cloud resources



### Flexibility & Extensibility

- Platform Independent
- Flexible Architecture
- Easily integrated with existing tools

# The main features of OpenNebula



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



# OpenNebula Capability

---

## For the Infrastructure Manager

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

## For the Infrastructure User

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

# OpenNebula Cloud Model

---

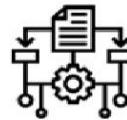
Embrace the power of Nebula Cloud's HPC & GPU Computing Resources



Flexible Pricing



Unified Billing



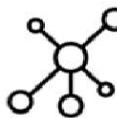
Multi-cloud



Proven Digital Commerce 1-Click Launch



Easy Scalability



Global Deployments



Secure Network



Managed Services



End to End SLA

# Open nebula Users

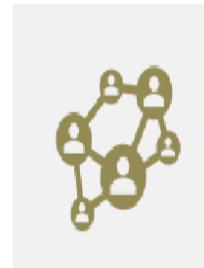
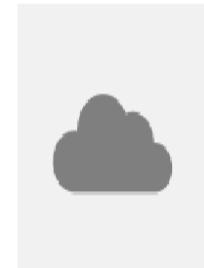
BUILDING ENTERPRISE CLOUDS IN MEDIUM AND LARGE DATA CENTERS



# Nebula Cloud – A Cutting Edge Digital Collaboration Platform

Experience how we transform the way enterprises run and manage global workforce

The platform will exponentially increase global workshare and interactive collaboration via a cutting-edge digital commerce platform which allows access to significant data and computing power from anywhere in the world, at any time, and on any device.



## Higher Efficiency

By increasing the pace of project delivery

## Global Collaboration

Increase global workshare and collaboration across time zones and disciplines.

## HPC & GPU Computing

High Performance Computing and GPU computing anytime, anywhere, on any device

## Higher ROI

Consumption based pricing and need based licensing lowers Total Cost of Ownership (TCO)

## Single Window Platform

It will allow clients and third parties simultaneously access the same environment

## Global Workshare

Higher Productivity and efficiency by enabling real-time collaboration between different contributors.

# Nebula Cloud Deployment Model

---

Solutions for every business need and situation

Nebula Cloud is one such way to experience virtual desktop infrastructure with preconfigured enterprise software and High Performance Computing (HPC) resources via its single window multi-cloud enabled DaaS platform. We have designed our solution to be simple and enterprise-ready.

Dedicated Virtual Machines  
for main users with critical  
usage requirements.

Single Desktop



Pooled Desktop

Assign a Single Desktop to  
a pool of users and  
efficiently manage the load  
of your virtual machines



Multi-Session Desktop



Target lower costs by  
pooling multi-session  
resources and reduce the  
number of virtual machines  
in your environment.

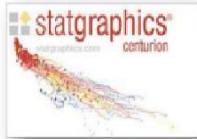
# Some of The Premier Partners

---

There is an opportunity for every customer success



CAD,CAM,CAE



BI & Analytics



CAD,CAM,CAE



GIS,3D,Earth Sciences



CAD,CAM,CAE,PLM,  
3D



Photogrammetry



GIS,BIM,3D



Design,Modelling



Earth Sciences



CAM,CAE,PLM,3D



BI, Big Data, ML,AI



BI, Data Sciences

\*over 200 curated products to support you in all aspects of your business

---

# Interoperability from the Cloud Provider perspective

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

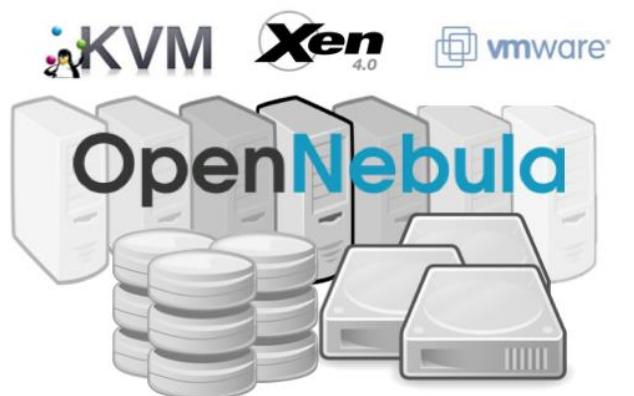
## Virtualization Drivers



## Configuration

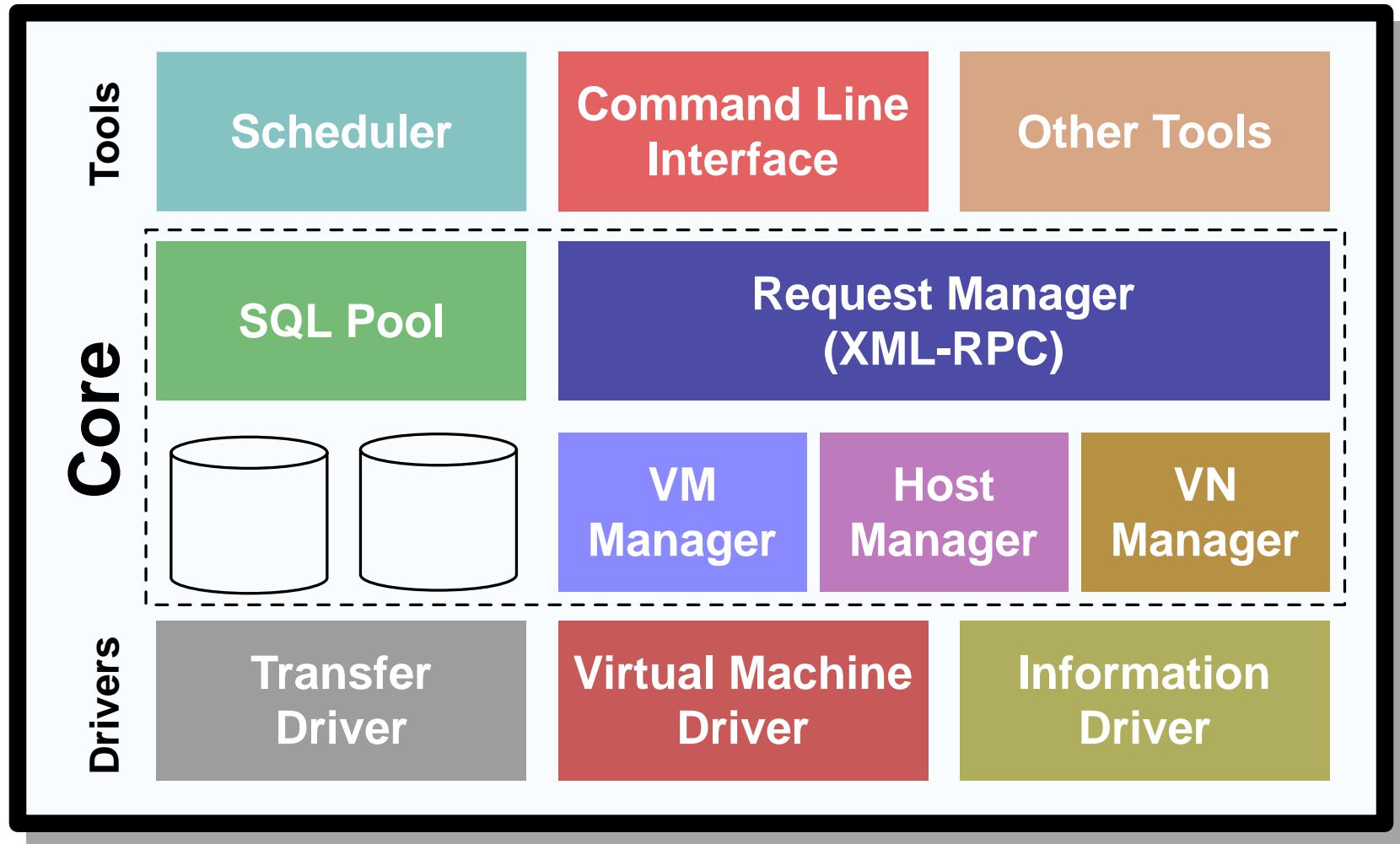


## Storage



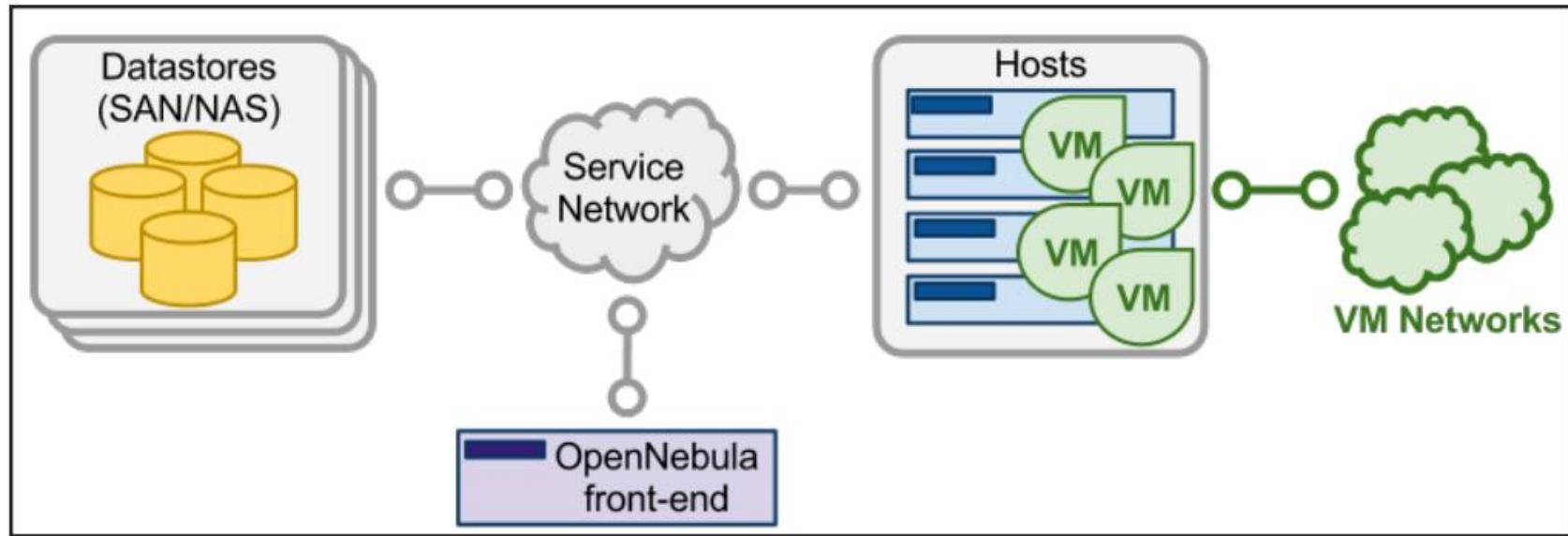
# Inside OpenNebula

# OpenNebula Architecture



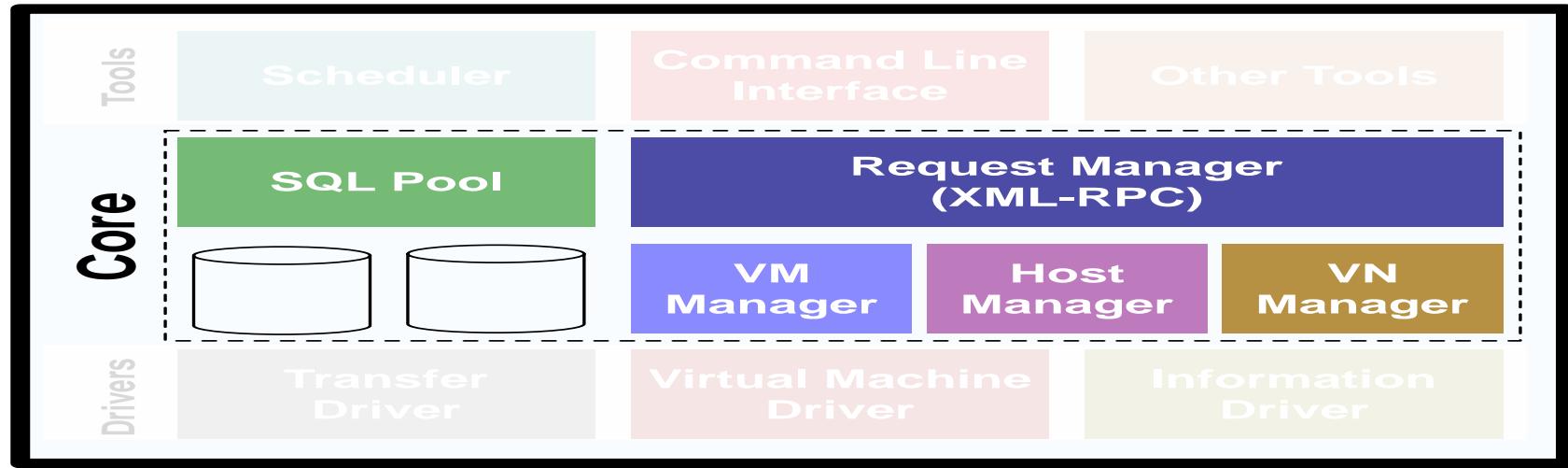
# Open Nebula Architecture Diagram

---



# The Core

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



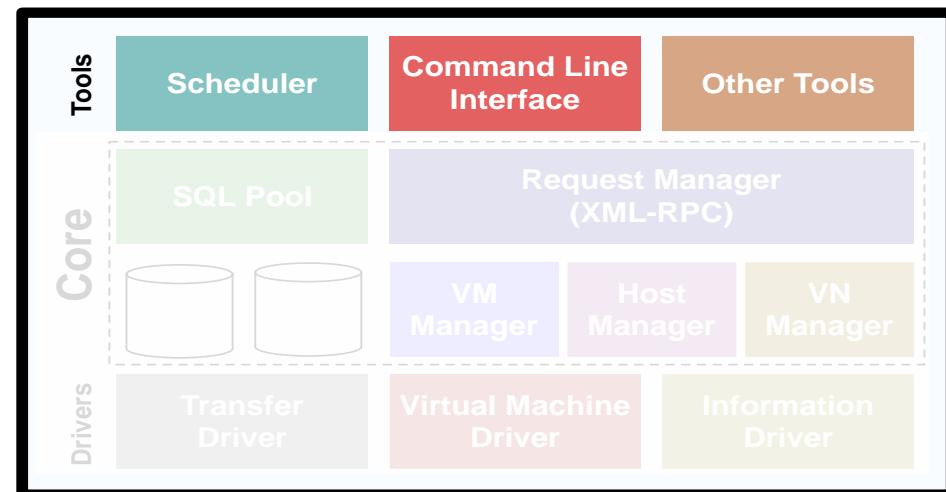
# The tools layer

## Scheduler:

- Searches for physical hosts to deploy newly defined VMs

## Command Line Interface:

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



# The drivers layer

---

Transfer Driver: Takes care of the images.

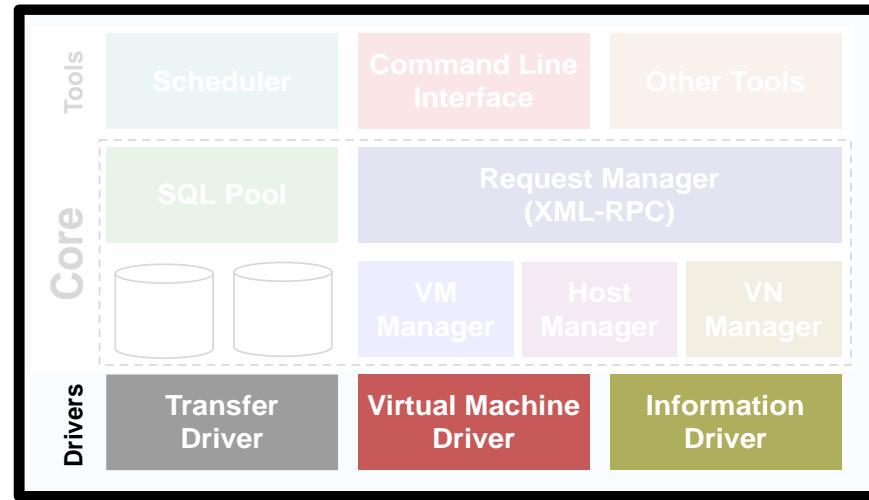
- cloning, deleting, creating swap image...

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

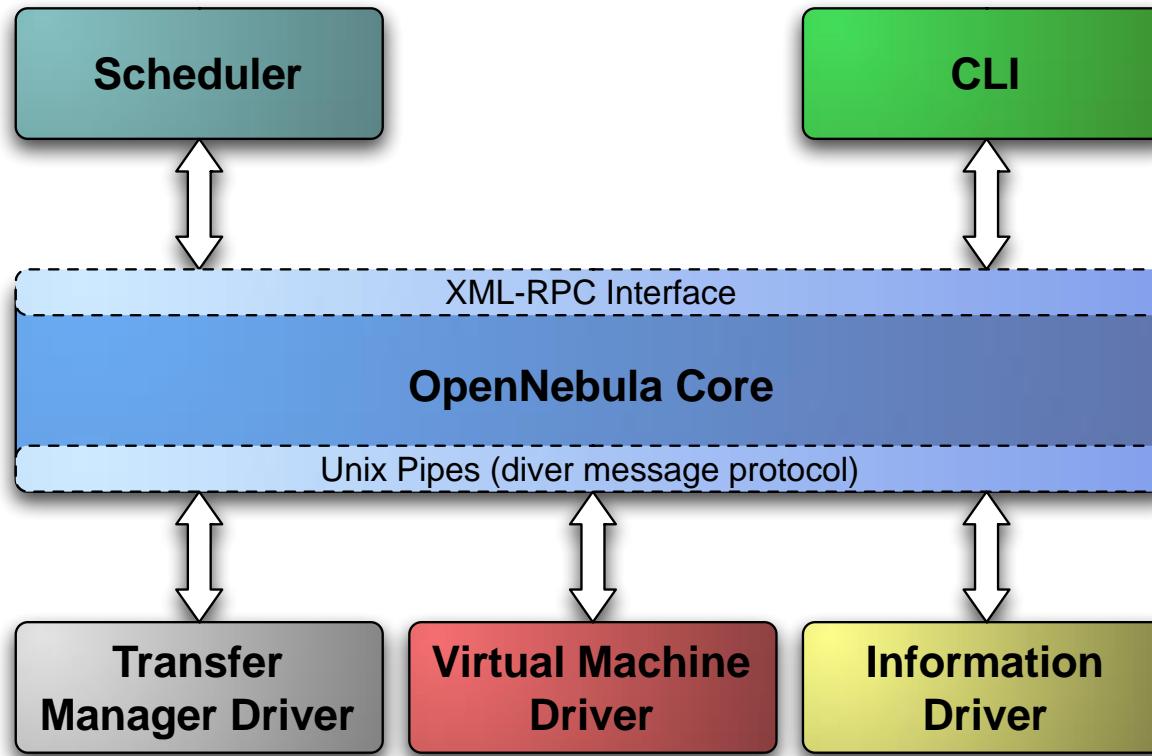
- deploy, shutdown, poll, migrate...

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

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



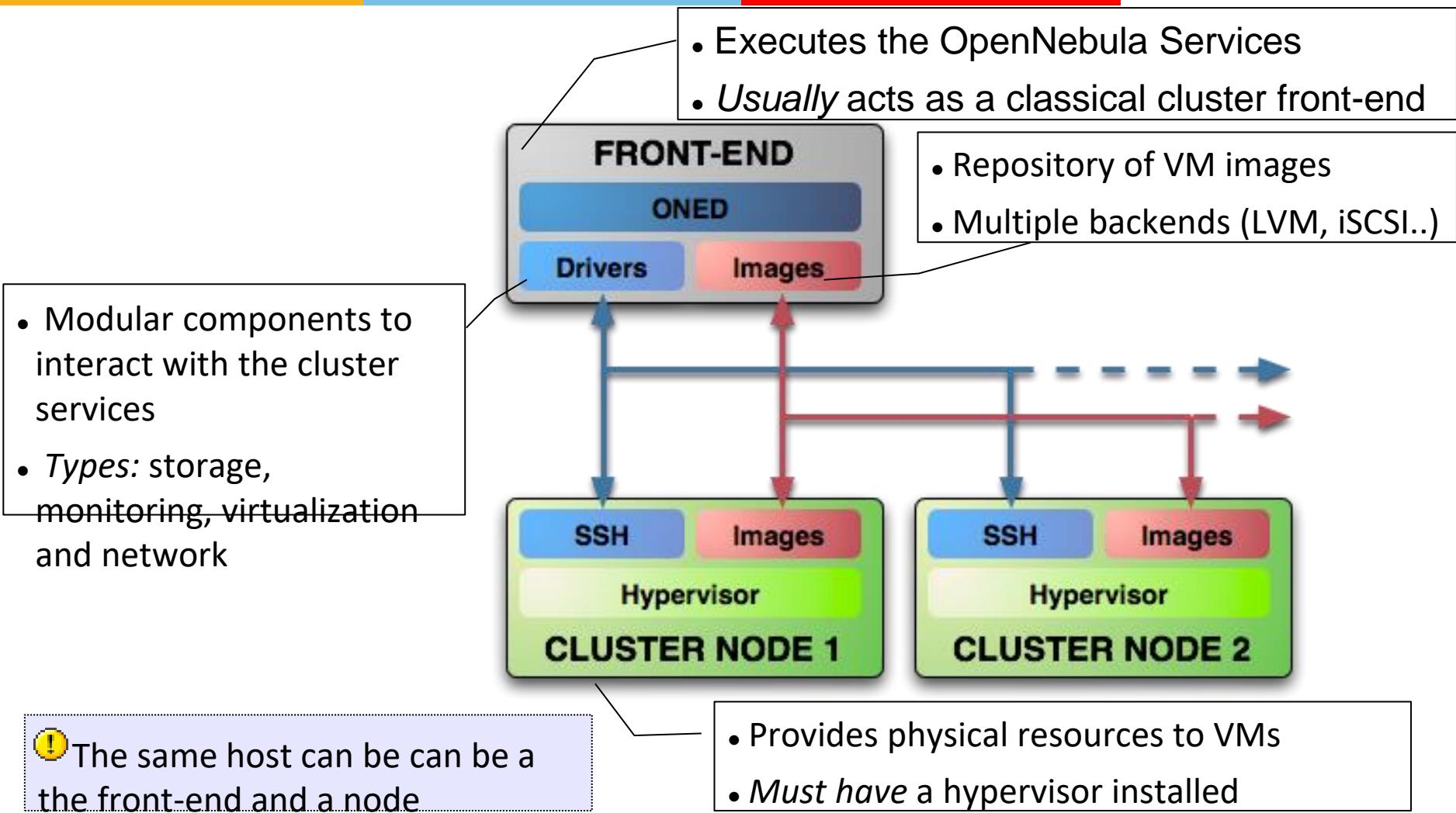
# Process separation



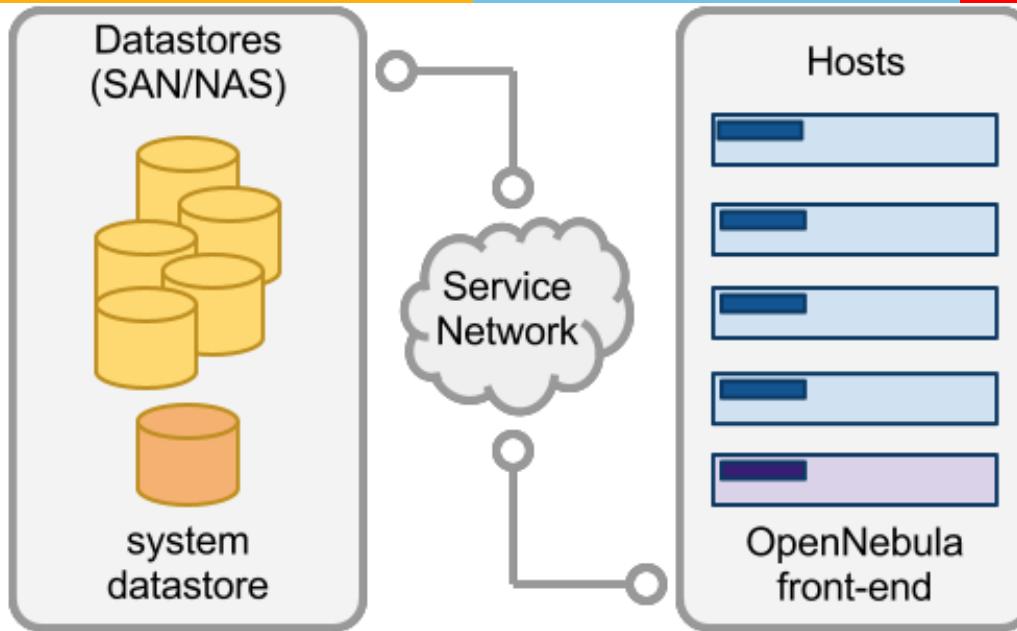
- Scheduler is a separated process, just like command line interface.
- Drivers are also separated processes using a simple text messaging protocol to communicate with OpenNebula Core Daemon (oned)

# Constructing a private cloud

# System Overview



# Complex Storage behind OpenNebula



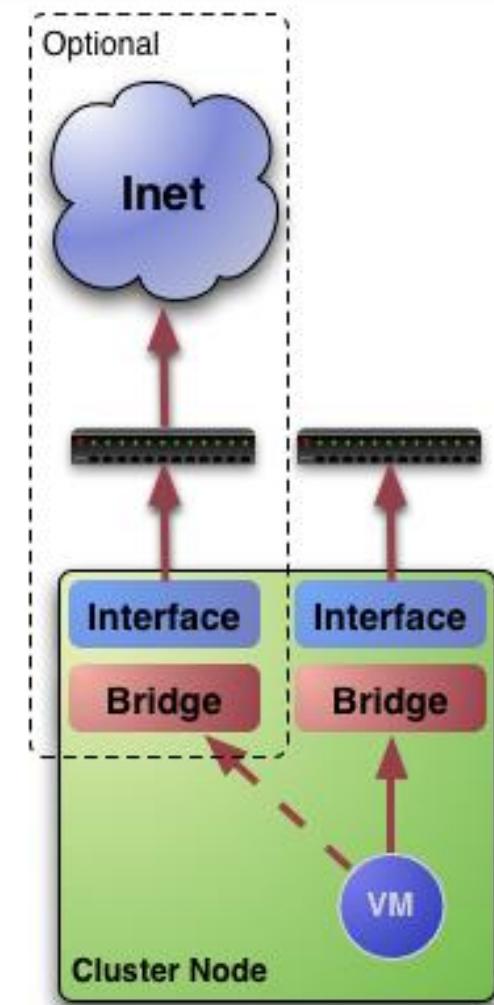
Datastore	Transfer Manager Drivers				
	shared	ssh	iscsi	qcow	vmware
System	OK	OK			
File-System	OK	OK		OK	
iSCSI			OK		
VMware	OK	OK			OK

Virtual machines and their images are represented as files

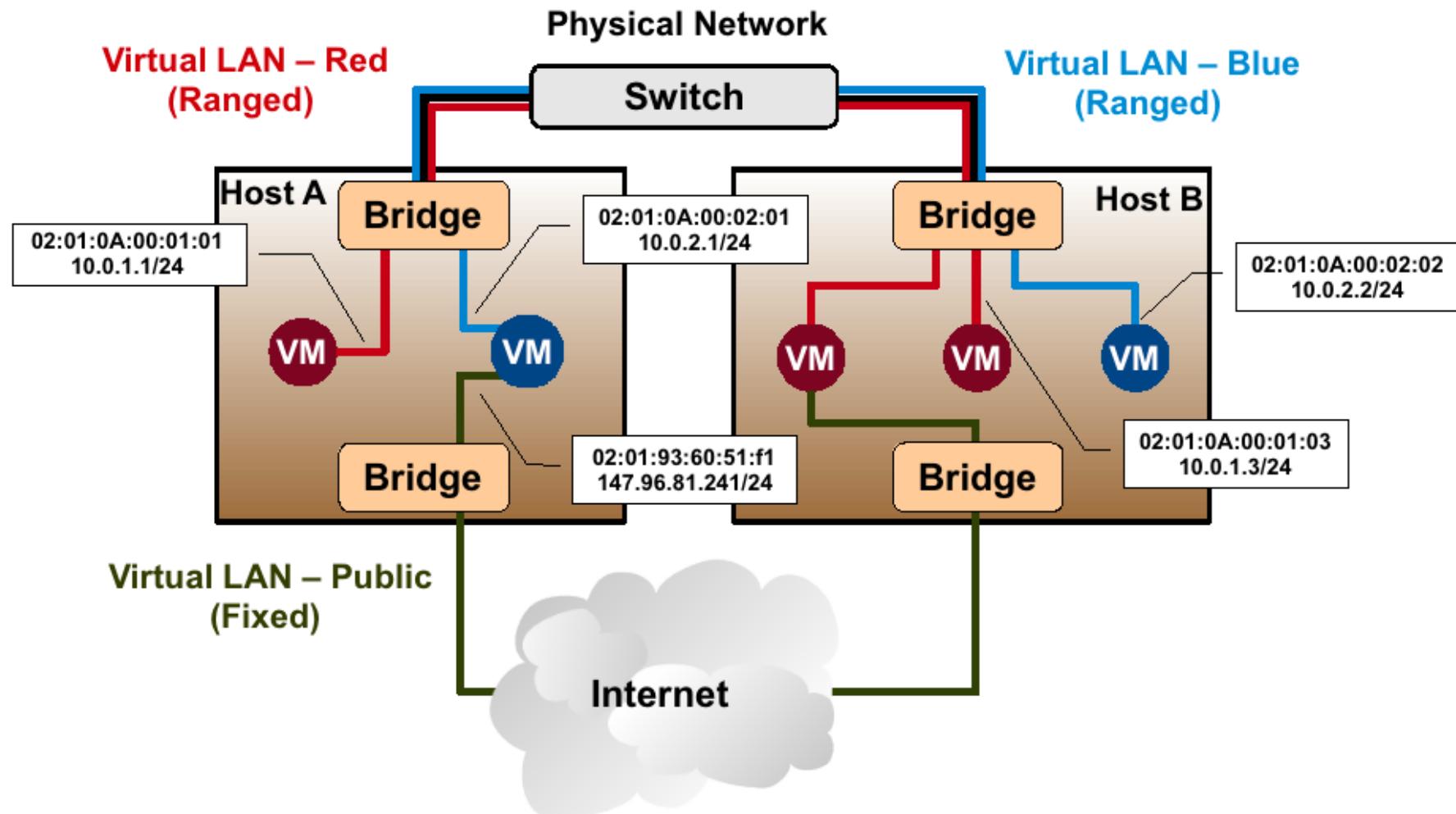
Virtual machines and their images are represented as block devices (just like a disk)

# Networking for private clouds

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



# Example network setup in a private cloud



# Virtual machines

# VM Description

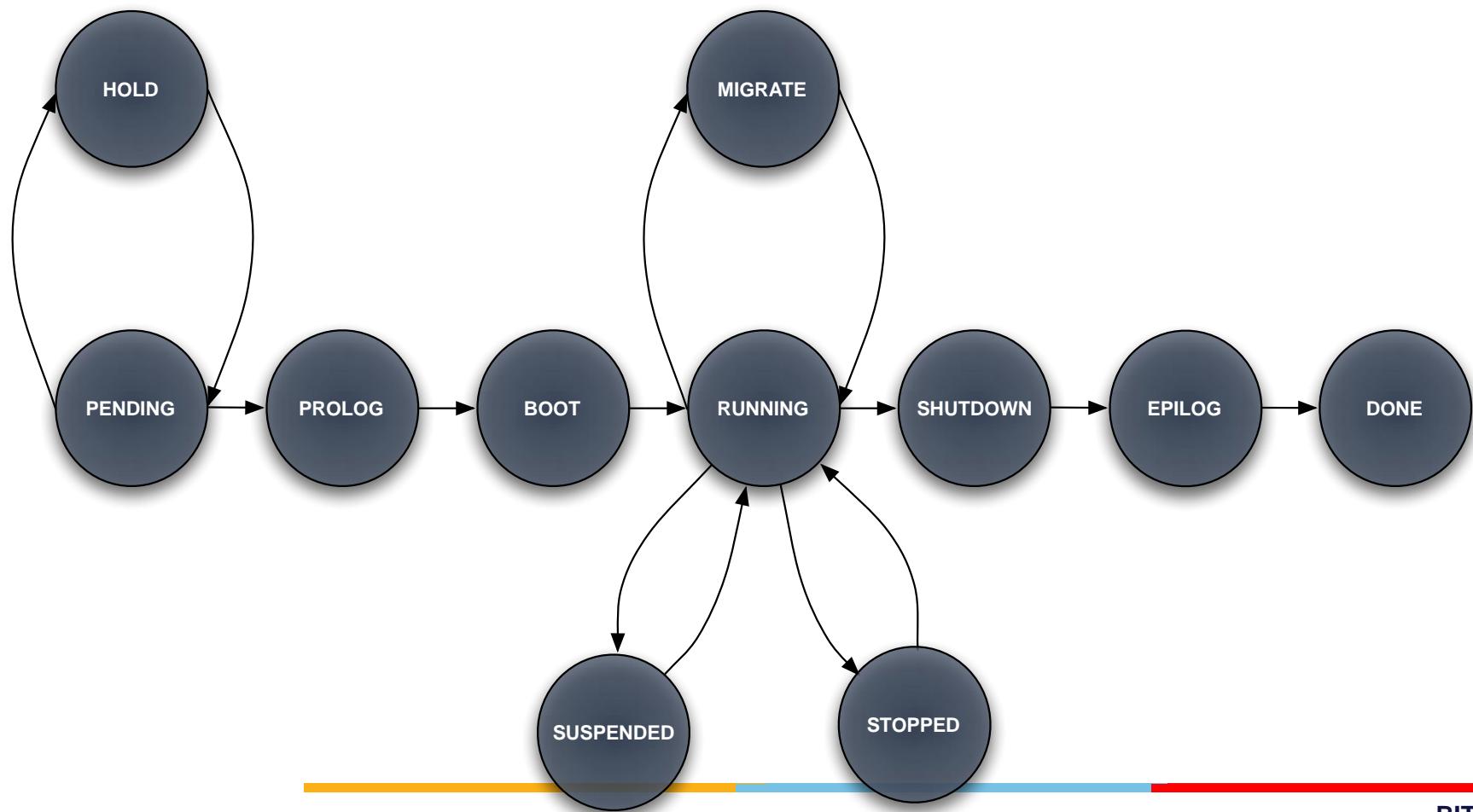
Option	Description
NAME	<ul style="list-style-type: none"><li>Name that the VM will get for description purposes.</li></ul>
CPU	<ul style="list-style-type: none"><li>Percentage of CPU divided by 100 required for the Virtual Machine.</li></ul>
OS (KERNEL, INITRD)	<ul style="list-style-type: none"><li>Path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

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

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

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

# VM States overview



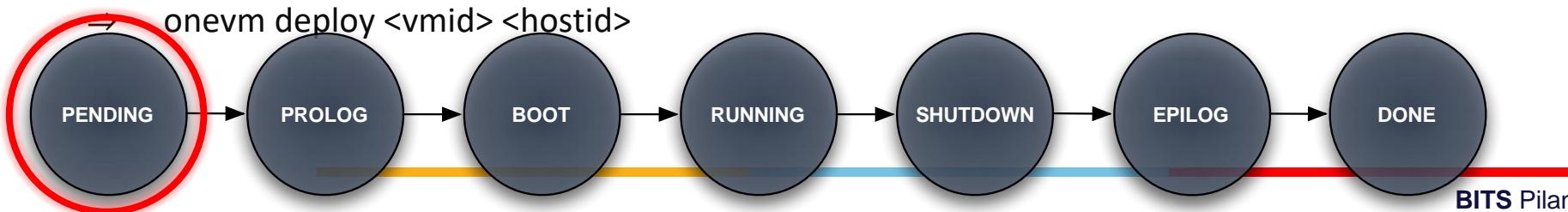
# Pending state

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

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

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

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

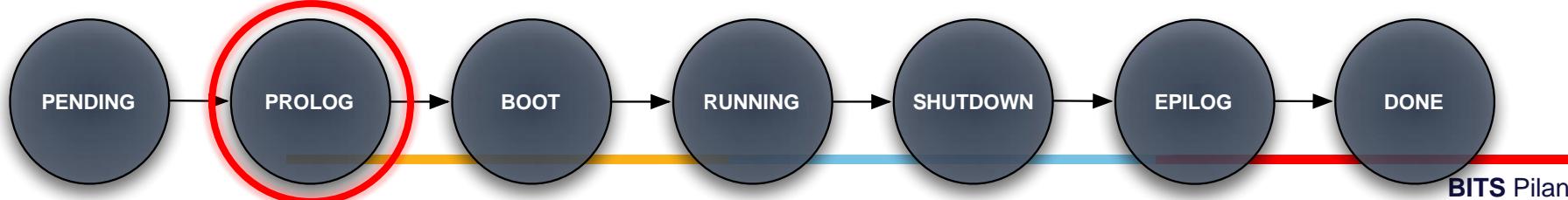


# Prolog state

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

Transfer actions:

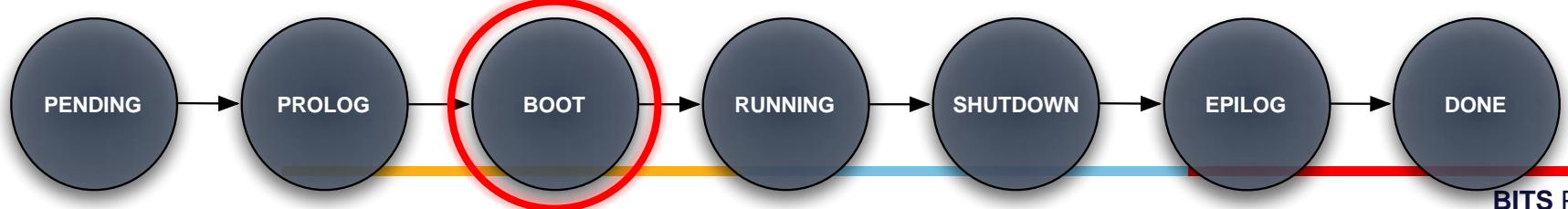
- **CLONE**: Makes a copy of a disk image file to be used by the VM. If Clone option for that file is set to false and the Transfer Driver is configured for NFS then a symbolic link is created.
- **MKSWAP**: Creates a swap disk image on the fly to be used by the VM if it is specified in the VM description.



# Boot state

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

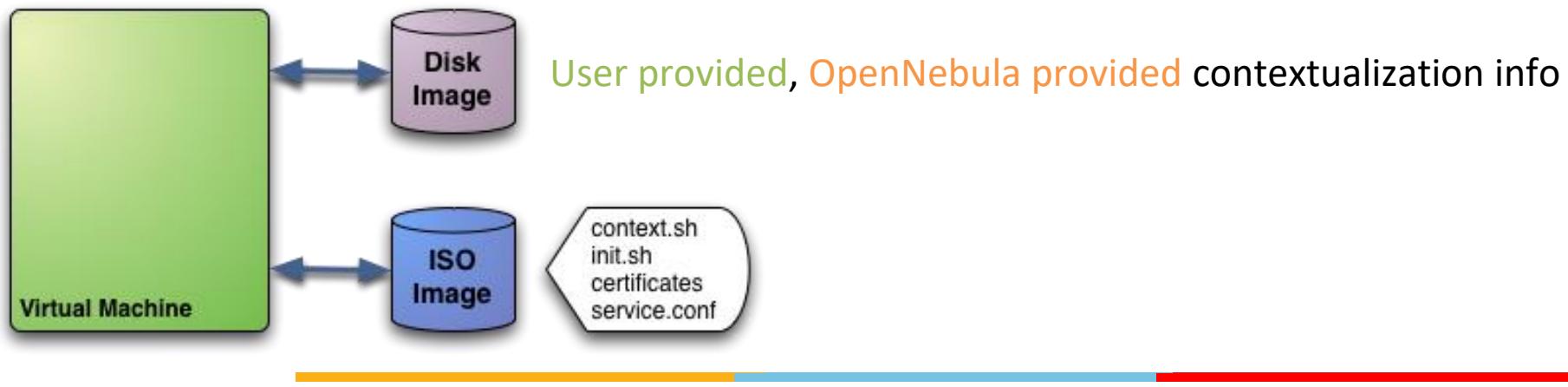
The VM will be in this state until deployment finishes or fails.



# Contextualization

The ISO image has the contextualization for that VM:

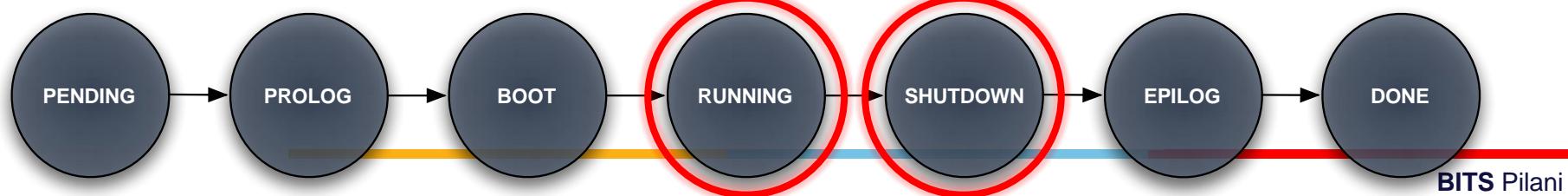
- **context**: contains configuration variables
- **init**: script called by VM at start to configure specific services
- **certificates**: directory that contains certificates for some service
- **service.conf**: service configuration



# Running and Shutdown states

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

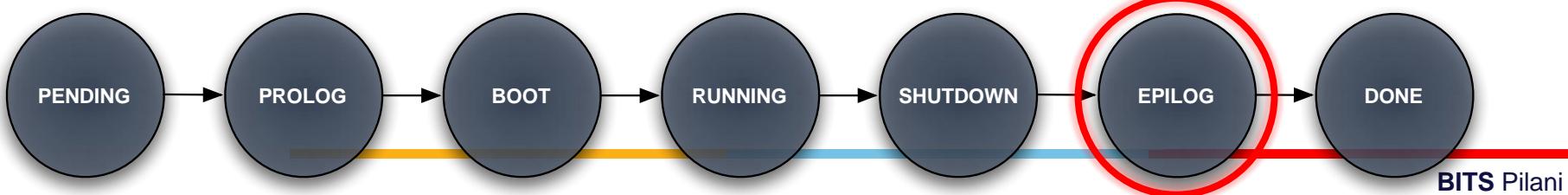
In SHUTDOWN state Virtual Machine Driver will send the shutdown command to the underlying virtual infrastructure.



# Epilog state

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

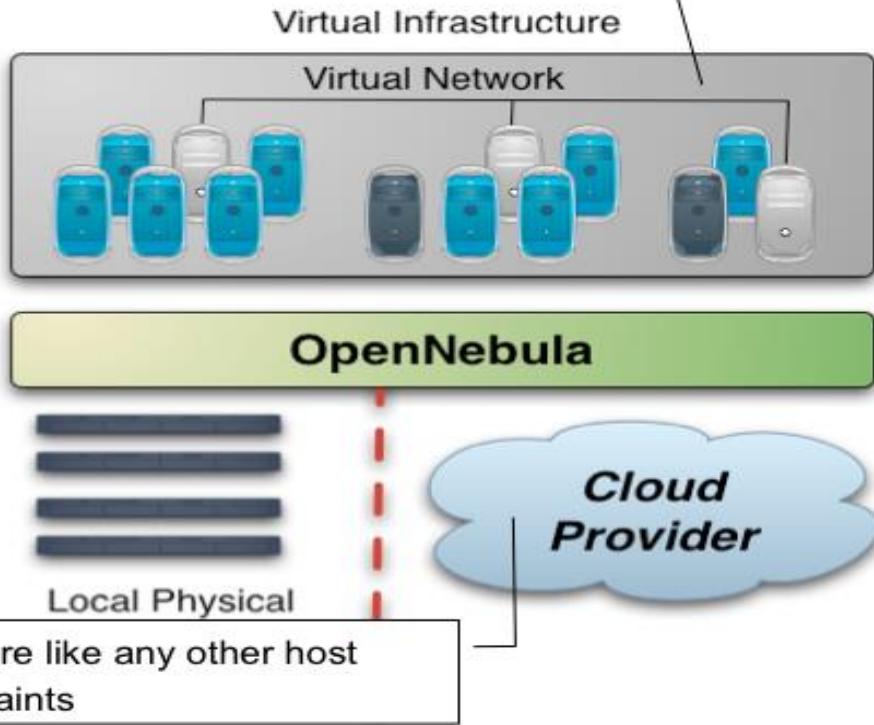
- Copy back the images that have **SAVE=yes** option.
- Delete images that were cloned or generated by **MKSWAP**.



# Hybrid cloud

# Overview

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



# Making an Amazon EC2 hybrid

---

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

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

# Using an EC2 hybrid cloud

---

Virtual Machines can be instantiated locally or in EC2

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

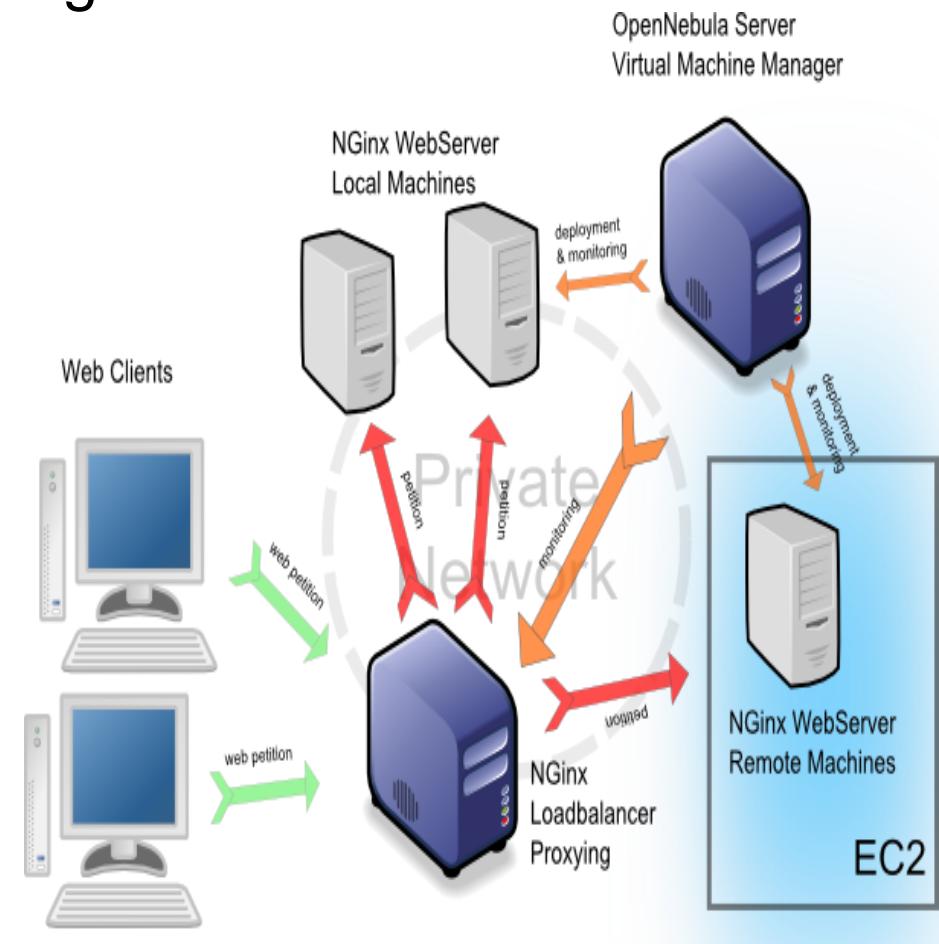
The EC2 counterpart of your VM (AMI\_ID) must be available for the driver account

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

# Hybrid cloud Use Case

## On-demand Scaling of Computing Clusters

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



# Open Source Cloud Services

**Table 1. Cloud Architectures Compared**

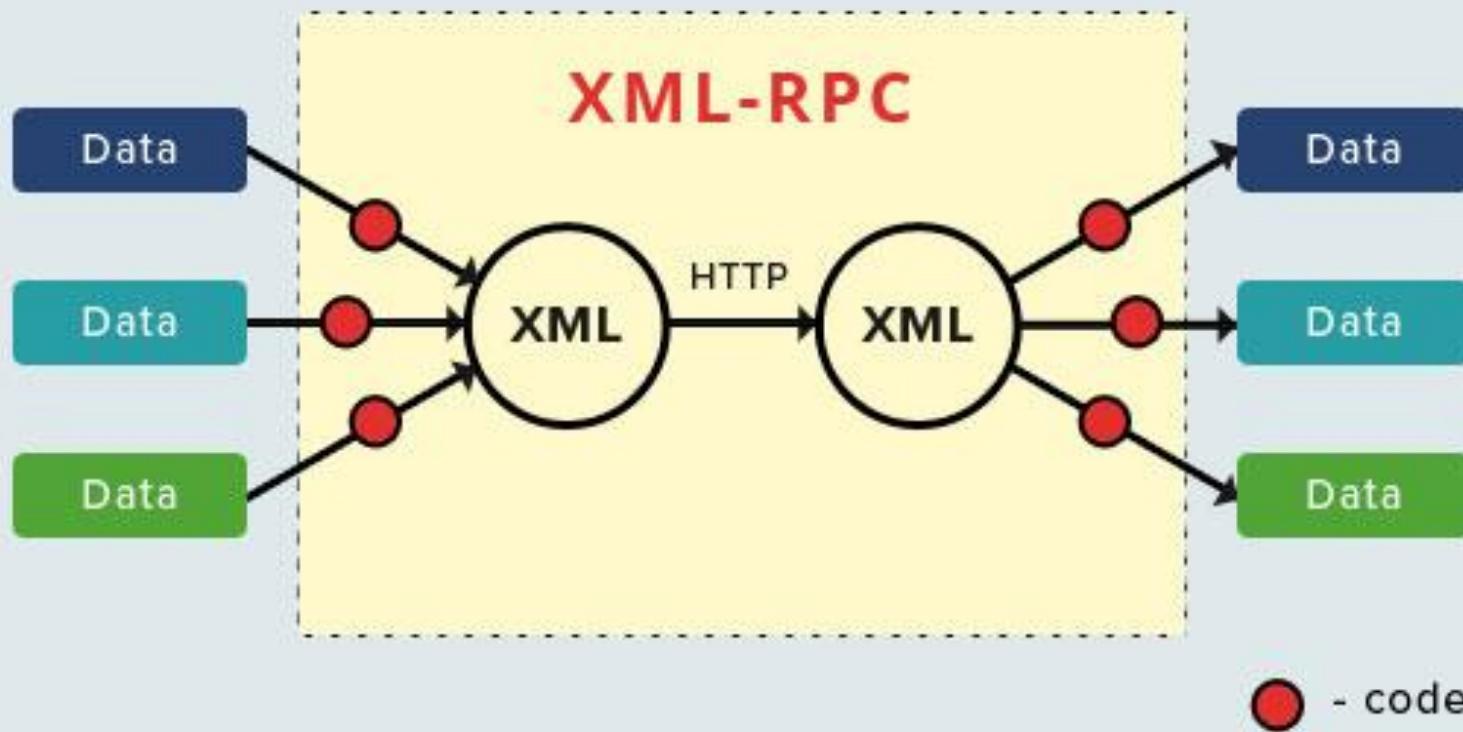
	<b>Eucalyptus</b>	<b>OpenNebula</b>	<b>Nimbus</b>
Disk Image Options	Options set by admin	In private cloud, most libvirt options left open.	Depends on configuration
Disk Image Storage	Walrus, which imitates Amazons S3	A shared file system, by default NFS, or SCP	Cumulus (recent update from GridFTP)
Hypervisors	Xen, KVM (VM Ware in non-open source)	Xen, KVM, VMware	Xen, KVM
Unique Features	User management web interface	VM migration supported	Nimbus context broker

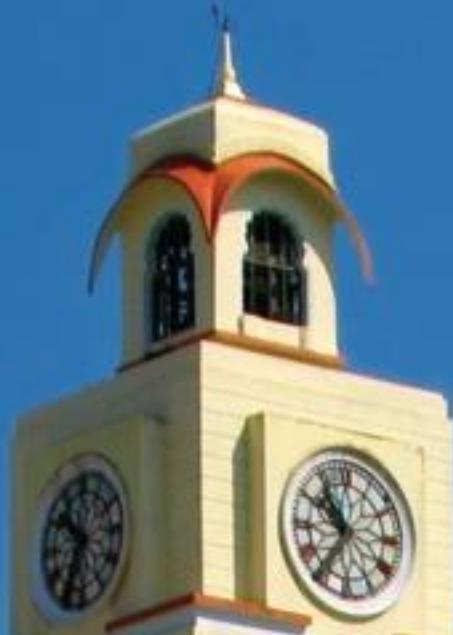
## Summary of Open Cloud Services

**Table 2. Summary of Results**

	Eucalyptus	OpenNebula	Nimbus
Philosophy	Mimic Amazon EC2	Private, highly customizable cloud	Cloud resources tailored to scientific researchers
Customizability	Some for admin, less for user	Basically everything	Many parts except for image storage and globus credentials
DHCP	On cluster controller	Variable	On individual compute node
Internal Security	Tight. Root required for many things.	Looser, but can be made more tight if needed.	Fairly tight, unless deploying a fully private cloud.
User Security	Users are given custom credentials via a web interface	User logs into head (unless optional front-end used)	Users x509 credential is registered with cloud
An Ideal Setting	Large group of machines for bunch of semi-trusted users	Smaller group of machines for highly trusted users	Deploy for less to semi-trusted users familiar with x509
Network Issues	dhcpd on cluster controller	Admin must set manually but has many options	dhcpd on every node and Nimbus assigns MAC

# XML-RPC communication





# Cloud Computing

## SEWP ZG527

**BITS** Pilani

## Capacity Management Challenges

---

- Cloud resource management .
  - Requires complex policies and decisions for multi-objective optimization.
  - It is challenging - the complexity of the system makes it impossible to have accurate global state information.
  - Affected by unpredictable interactions with the environment, e.g., system failures, attacks.
  - Cloud service providers are faced with large fluctuating loads which challenge the claim of cloud elasticity.
- The strategies for resource management for IaaS, PaaS, and SaaS are different.

## Capacity Planner Goal



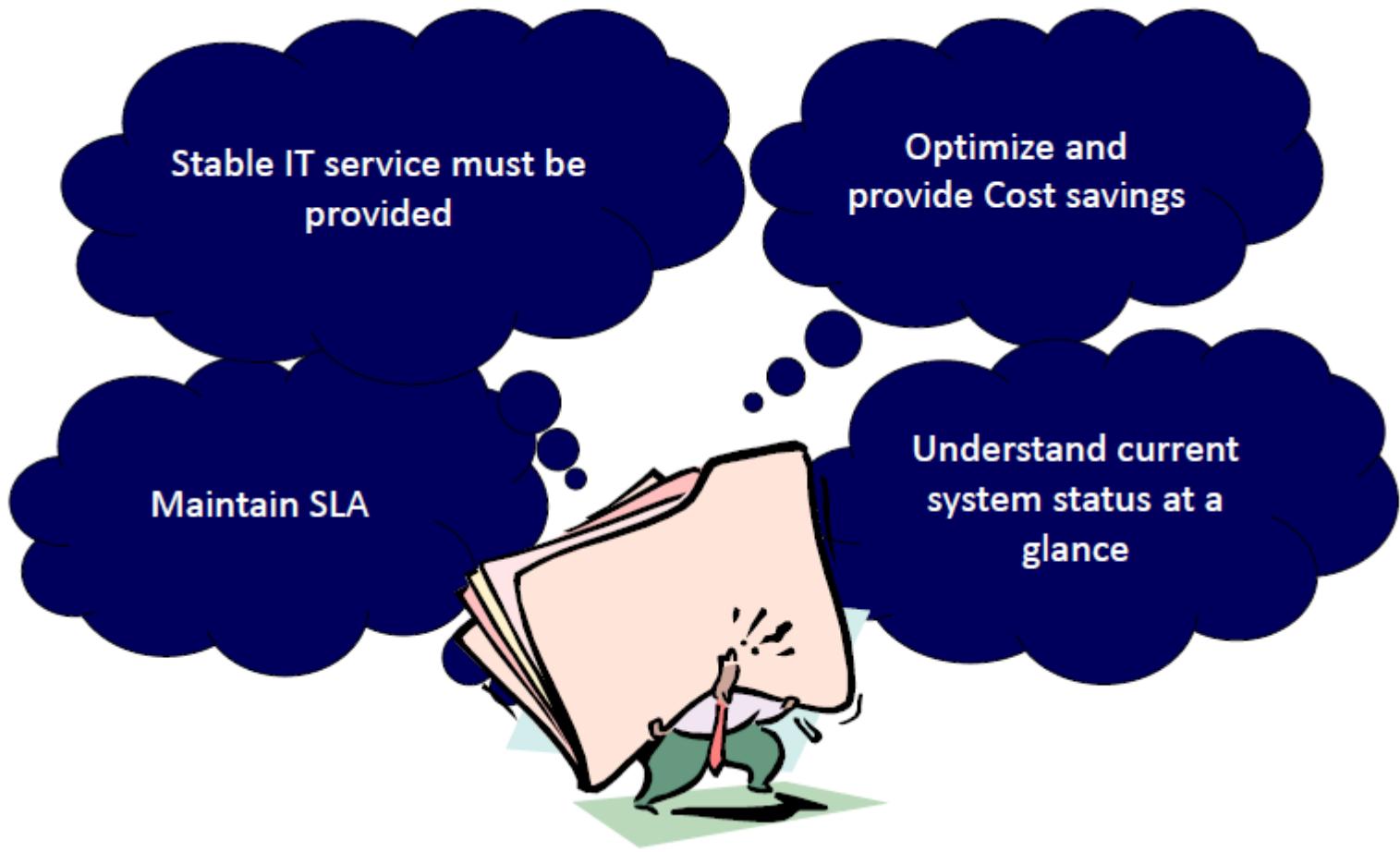
The goal of capacity planners is to identify significant & vital resources that have resource ceiling & add more resources to move the restricted access to higher levels of demand.

Network capacity is one of the hardest factors to resolve & the performance of the network is affected by I/O of the network at the server & network traffic from cloud to ISPs (Internet Service Providers).

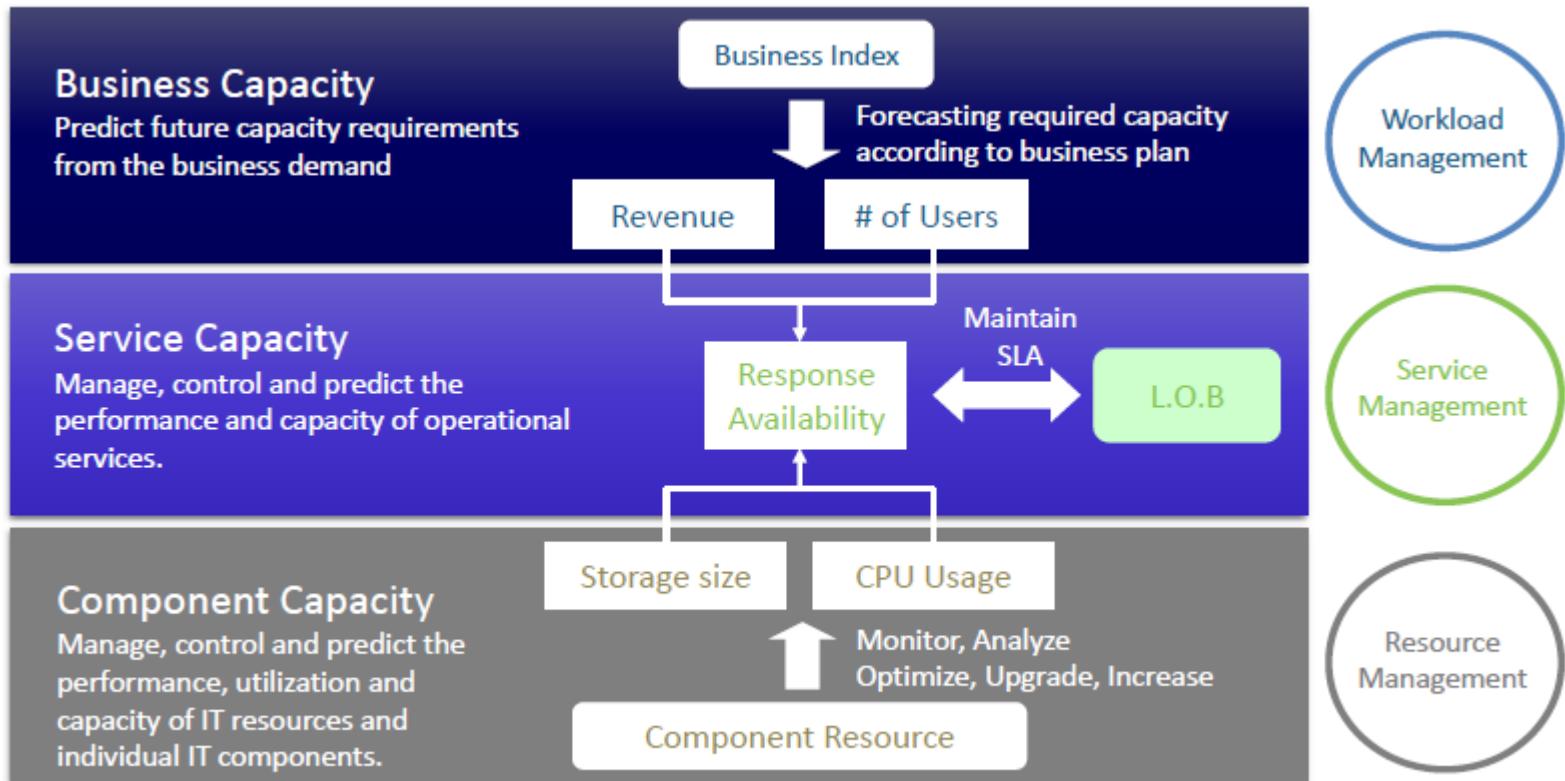
Capacity planners try to find the solution to meet future demands on a system by providing additional capacity to fulfill those demands. Capacity planning & system optimization are two both different concepts, and you mustn't mix them as one. Performance & capacity are two different attributes of a system. Cloud 'capacity' measures & concerns about how much workload a system can hold whereas 'performance' deals with the rate at which a task get performed

## Why Capacity Management

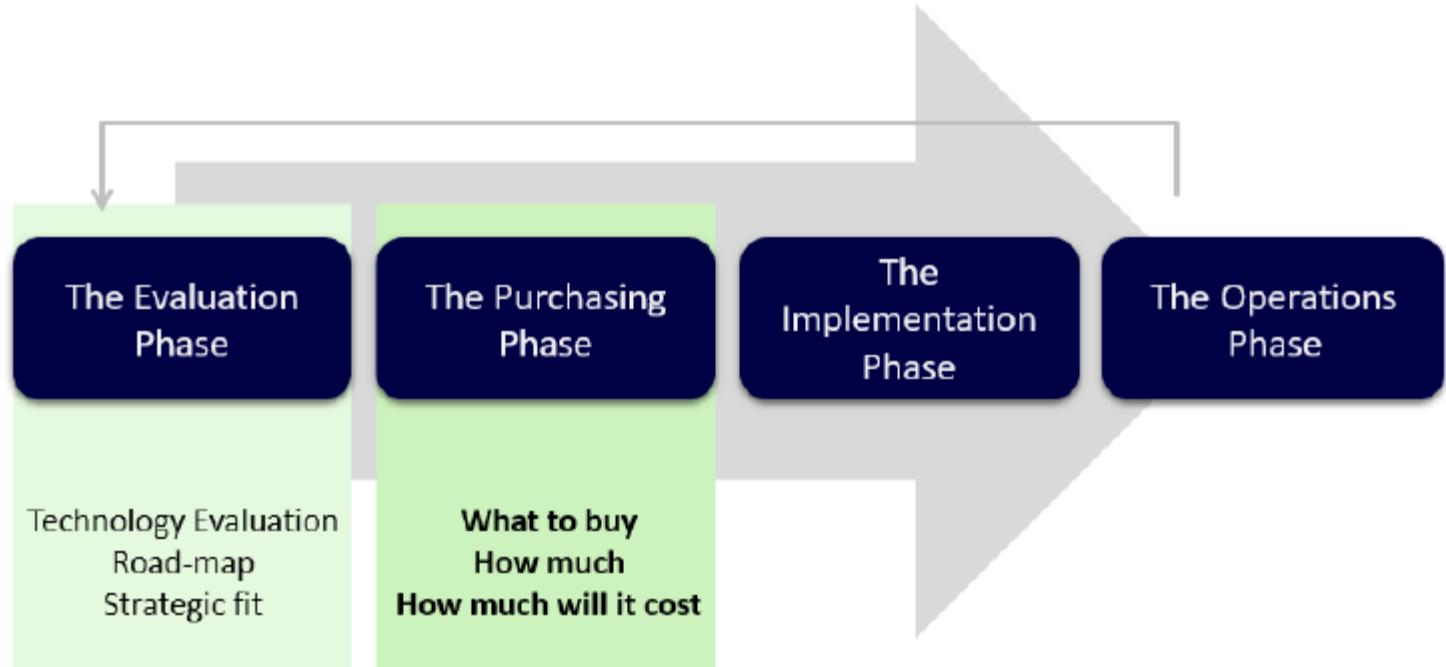
---



# Capacity Management Building Blocks

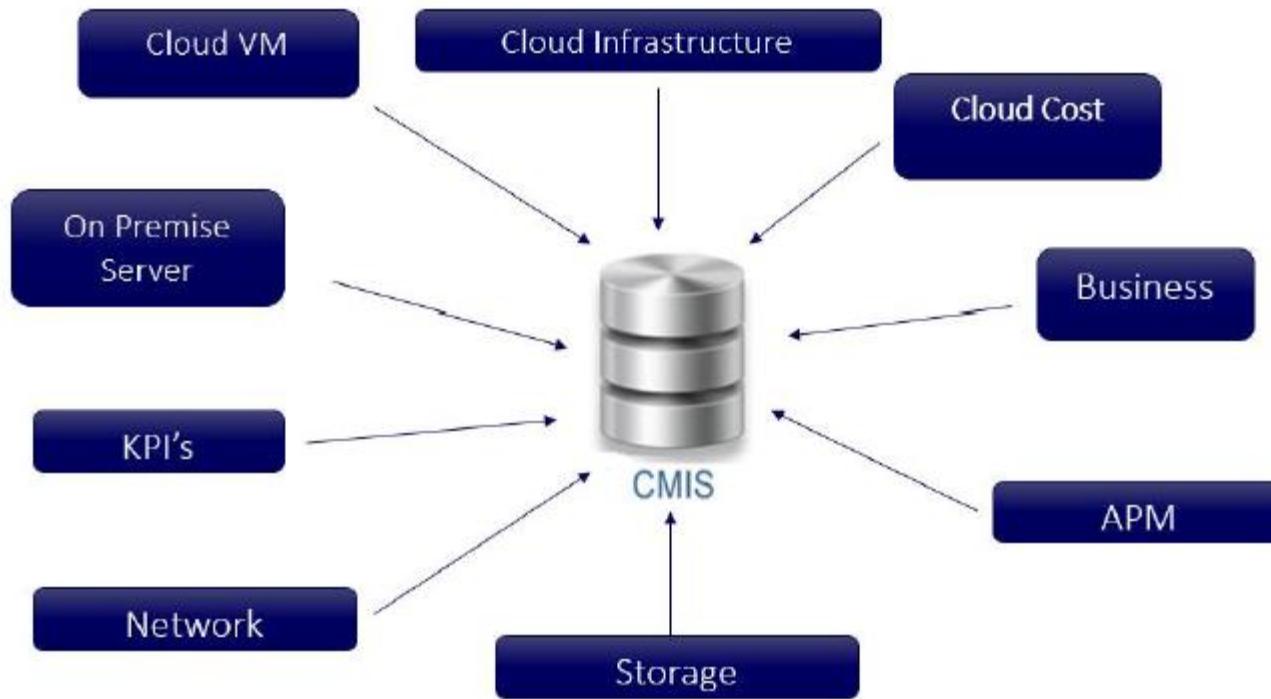


# Cloud Deployment Model



# Cloud Management

## Capacity Management Information System (CMIS)



## Capacity Resource Management Policies

---

1. Admission control → prevent the system from accepting workload in violation of high-level system policies.
  2. Capacity allocation → allocate resources for individual activations of a service.
  3. Load balancing → distribute the workload evenly among the servers.
  4. Energy optimization → minimization of energy consumption.
  5. Quality of service (QoS) guarantees → ability to satisfy timing or other conditions specified by a Service Level Agreement.
-

## Capacity Planning Steps

---

- Determine the distinctiveness of the present system.
  - Determine the working load for different resources in the system such as CPU, RAM, network, etc.
  - Load the system until it gets overloaded; & state what's requiring to uphold acceptable performance.
  - Predict the future based on older statistical reports & other factors.
  - Deploy resources to meet the predictions & calculations.
  - Repeat step (i) through (v) as a loop.
-

# Capacity Management Framework

M  
a  
n  
a  
g  
e

t  
h  
e

P  
r  
o  
c  
e  
s  
s

**Establish Capacity Management Framework:** Based on the business and IT strategy and the architectural models, guidelines and a framework for capacity management will be developed.

## Model and Size Capacity Requirements:

- Modeling involves performance and capacity prediction through estimation, trend analysis, analytical modeling, simulation modeling and benchmarking. Application sizing is a technique that predicts the capacity solution required to meet service level requirements for response times, throughput, and batch elapsed times.

## Monitor, Analyze, and Report Capacity Usage:

- Monitors should be established on all the components and for each of the services. The data should be analyzed using, wherever possible, expert systems to compare usage levels against thresholds. The results of the analysis should be included in reports, and recommendations made as appropriate.

## Supervise Tuning and Capacity Delivery:

- Outputs from monitoring, analyzing, and reporting activities are examined and actions to tune individual resources or to re-balance the available capacity are planned and initiated through Change and Release Management.

## Produce and Maintain Capacity Plan:

- The objective of this activity is to develop, maintain, test and revise alternative approaches in satisfying various enterprise-shared resource requirements. It delivers the capacity plan that addresses the customer's resource requirements.

## Evaluate Capacity Management Process Performance:

Measurements include the definition, collection of measurements, analysis, review and reporting for Capacity Management.

## Implementation Capacity Resource Management Policies

---

- Control theory → uses the feedback to guarantee system stability and predict transient behavior.
- Machine learning → does not need a performance model of the system.
- Utility-based → require a performance model and a mechanism to correlate user-level performance with cost.
- Market-oriented/economic → do not require a model of the system, e.g., combinatorial auctions for bundles of resources.

## Trade offs

---

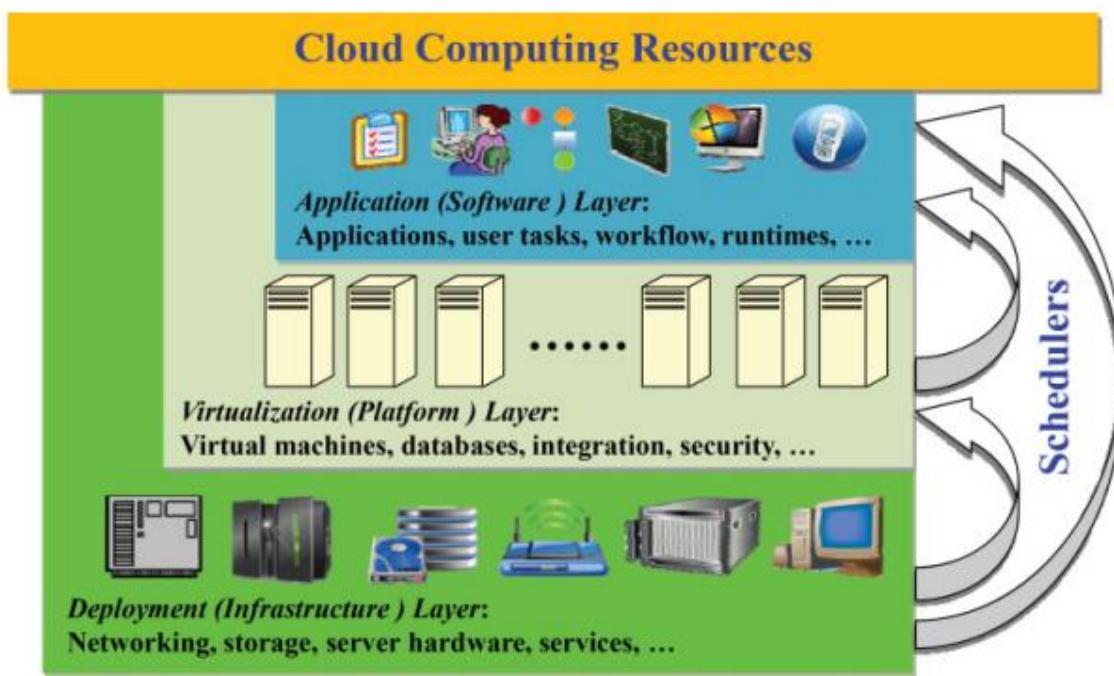
- To reduce cost and save energy we may need to concentrate the load on fewer servers rather than balance the load among them.
- We may also need to operate at a lower clock rate; the performance decreases at a lower rate than does the energy.

CPU speed (GHz)	Normalized energy (%)	Normalized performance (%)
0.6	0.44	0.61
0.8	0.48	0.70
1.0	0.52	0.79
1.2	0.58	0.81
1.4	0.62	0.88
1.6	0.70	0.90
1.8	0.82	0.95
2.0	0.90	0.99
2.2	1.00	1.00

- The main components of a control system:
  - The inputs → the offered workload and the policies for admission control, the capacity allocation, the load balancing, the energy optimization, and the QoS guarantees in the cloud.
  - The control system components → *sensors* used to estimate relevant measures of performance and *controllers* which implement various policies.
  - The outputs → the resource allocations to the individual applications.

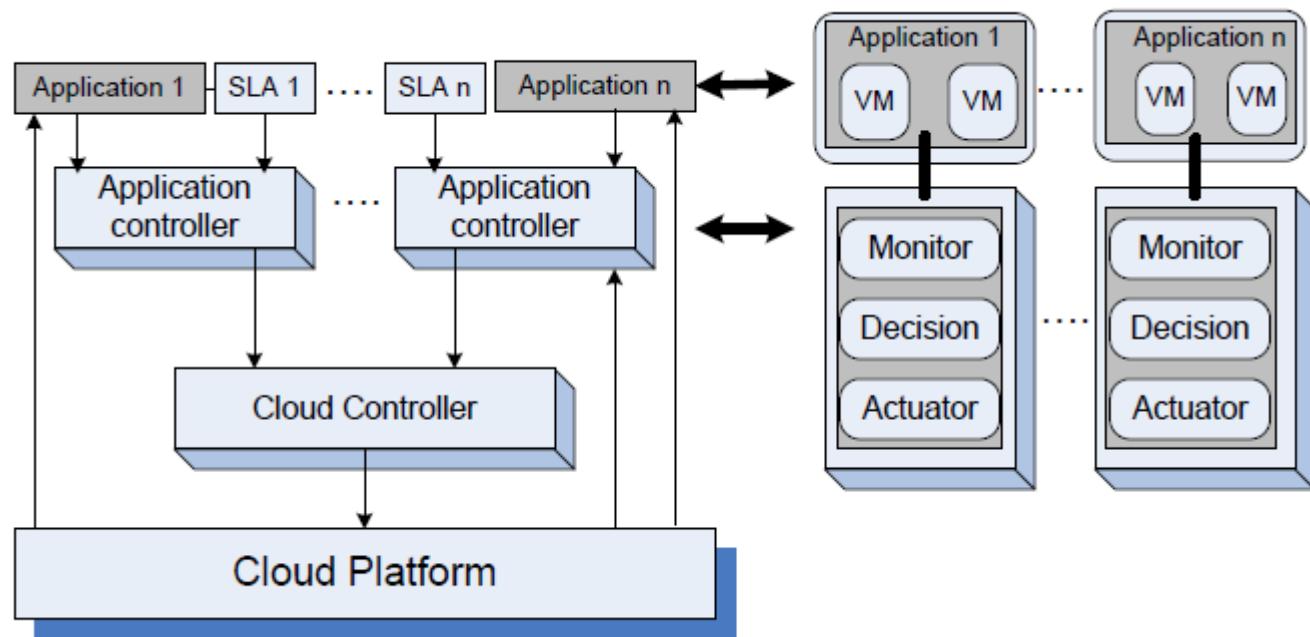
- Control granularity → the level of detail of the information used to control the system.
  - Fine control → very detailed information about the parameters controlling the system state is used.
  - Coarse control → the accuracy of these parameters is traded for the efficiency of implementation.
- The controllers use the feedback provided by sensors to stabilize the system. Stability is related to the change of the output.
- Sources of instability in any control system:
  - The delay in getting the system reaction after a control action.
  - The granularity of the control, the fact that a small change enacted by the controllers leads to very large changes of the output.
  - Oscillations, when the changes of the input are too large and the control is too weak, such that the changes of the input propagate directly to the output.

# Cloud Resource Scheduling

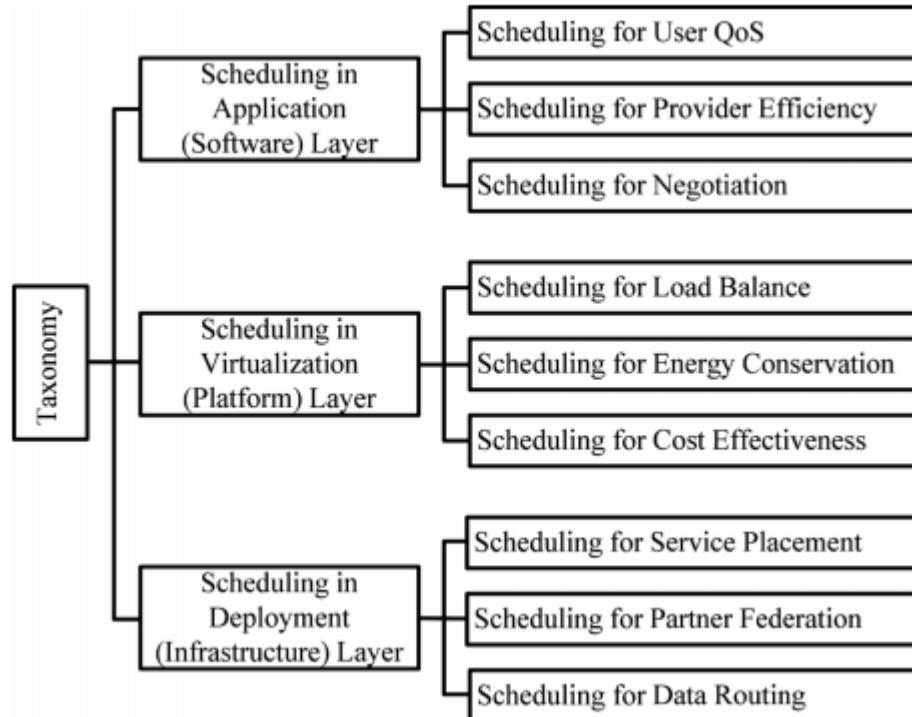


- Scheduling in the application layer is to schedule the virtual or physical resources to support software and user applications, tasks, and workflows, etc., with optimal QoS and efficiency.
- Scheduling in the virtualization layer focuses on mapping virtual resources onto physical resources with optimal load balance, energy conservation, etc.
- Scheduling in the deployment layer, which also attracts worldwide attention, is concerned with optimal and strategic infrastructure, outsourcing, service placement, multicloud centers, partnering, data routing and application migration, etc

## Cloud Control Level



# Cloud Resource Scheduling



## Design Considerations

---

- Is it beneficial to have two types of controllers:
  - application controllers → determine if additional resources are needed.
  - cloud controllers → arbitrate requests for resources and allocates the physical resources.
- Choose fine versus coarse control.
- Dynamic thresholds based on time averages better versus static ones.
- Use a high and a low threshold versus a high threshold only.

## Proportional Thresholding's

---

### ■ Algorithm

- Compute the integral value of the high and the low threshold as averages of the maximum and, respectively, the minimum of the processor utilization over the process history.
- Request additional VMs when the average value of the CPU utilization over the current time slice exceeds the high threshold.
- Release a VM when the average value of the CPU utilization over the current time slice falls below the low threshold.

### ■ Conclusions

- Dynamic thresholds perform better than the static ones.
- Two thresholds are better than one.

## Resource Bundlings

---

- ◆ Resources in a cloud are allocated in **bundles**.
- ◆ Users get maximum benefit from a specific combination of resources: CPU cycles, main memory, disk space, network bandwidth, and so on.
- ◆ Resource bundling complicates traditional resource allocation models and has generated an interest in economic models and, in particular, in **auction algorithms**.
- ◆ The bidding process aims to optimize an objective function  $f(x,p)$ .
- ◆ In the context of cloud computing, **an auction is the allocation of resources to the highest bidder**.

# Capacity Management Capacity Journey

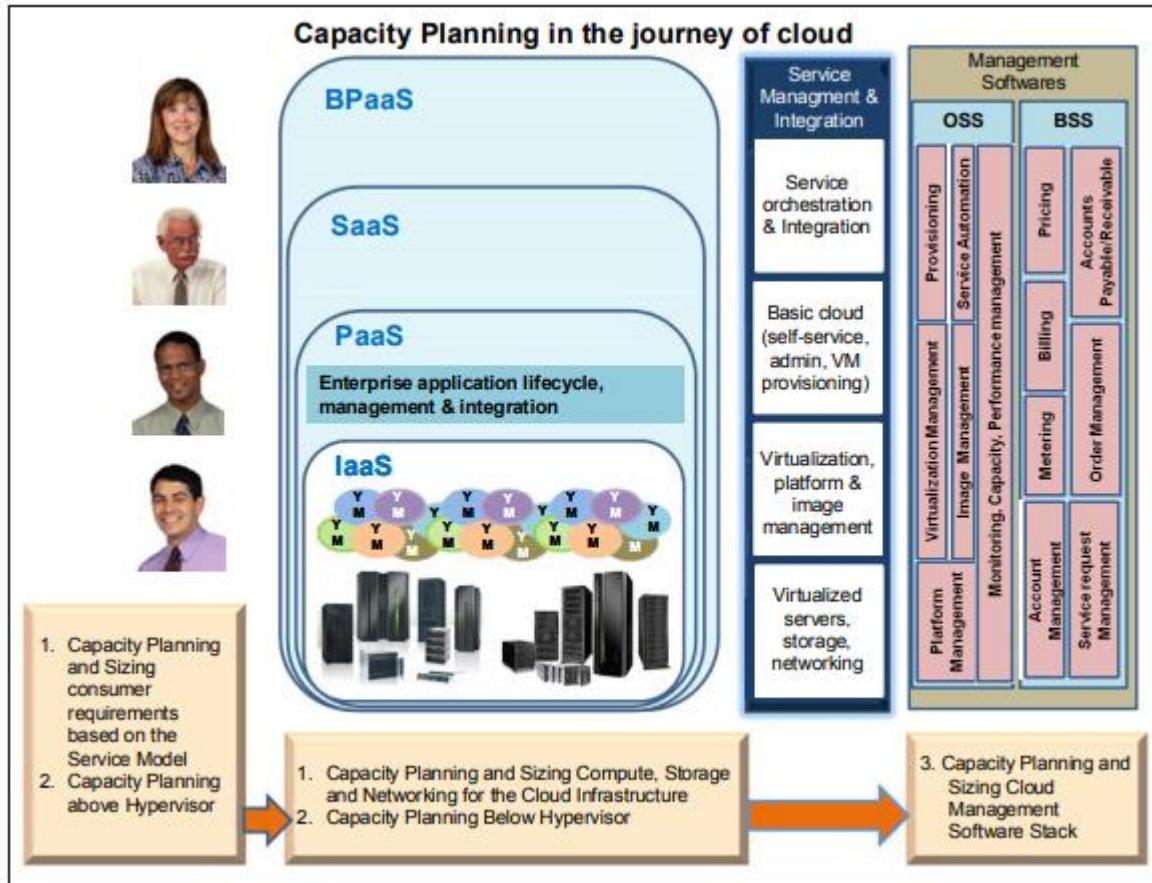


Figure 5.2 Capacity planning for a cloud journey

## Capacity Modelling

Capacity modeling involves categorization and characterization of entities involved in the end-to-end cloud infrastructure. There are many factors that need to be correlated to come up with the correct estimation model to provide optimal service to the consumers. Historical and live data need to be analyzed to forecast future requirements and to identify patterns to service different workloads efficiently. This is applicable for both the cloud consumer and provider:

1. Identify the tenancy model (single tenancy, multi-tenancy model).
2. Identify the performance characterization and volumetrics of the application with respect to tenancy model.
3. Identify the workload deployment type (development, test, and production).
4. Identify the correct servers and platform for your application requirements.
5. Analyze the CPU, memory, storage, and network requirements (minimum, average, and maximum).

## Cloud Application baselines

---

The first thing that strikes in mind while dealing with the business issue is the system's capacity or working load as a measurable quantity over time, since many developers build their cloud-based applications & websites based on the LAMP. The full-form is extracted below:

Linux - operating system

Apache - Apache Software Foundation's Web-server

MySQL - database server

PHP - Hypertext Preprocessor

The above four technologies are open-source although the distribution may vary from cloud to cloud. There are other slight variations of the LAMP that are available for development. These are:

OpAMP (OpenBSD Apache MySQL PHP)

SAMP (Solaris Apache MySQL PHP)

WAMP (Windows Apache MySQL PHP)

There are two important work-load matrices in the LAMP system. These are:

Page view: is the number of hits on a website & is measured in hits per second

Transactions: is measured by transactions per second and is the number of queries the database server completes per second

## Load Test

---

- Server administrator checks for servers under load for system metrics to give capacity planners enough information to do significant capacity planning. Capacity planners should know about the increase in load to the system. Load-testing needs to query the following questions:
- What is the optimum load that the system can support?
- What system blocks the current system & limits the system's performance?
- Can the configuration be altered in the server to use capacity?
- How will the server react concerning performance with other servers having different characteristics?



# Cloud Computing

## SEWP ZG527

**BITS** Pilani

# High Availability

---

High availability (HA) is the elimination of single points of failure to enable applications to continue to operate even if one of the IT components it depends on, such as a server, fails. IT professionals eliminate single points of failure to ensure continuous operation and uptime.

## **Highly Available Systems Incorporate Five Design Principles:**

- They automatically failover to a redundant system to pick up an operation when an active component fails. This eliminates single points of failure.
- They can automatically detect application-level failures as they happen, regardless of the causes.
- They ensure no amount of data loss during a system failure.
- They automatically and quickly failover to redundant components to minimize downtime.
- They provide the ability to manually failover and fallback to minimize downtime during planned maintenance.

# High Availability

---

- **Architect a High Availability Cloud Infrastructure**
  - How to design high available infrastructure for cloud
  - Architect hardware infrastructure to reduce unplanned outage
  - Design system architecture to minimize the planned/unplanned outage
  - Use configuration and implementation best practices for HA
  - Minimize downtime during system migration
  - Establish the pre-active real time monitoring system

## Steps to achieve high availability

- **Build for server failure**
- **Build for zone failure**
- **Build for Cloud failure**
- **Automating and testing**

# High Availability

---

The following elements help you implement highly available systems:

- **Redundancy**—ensuring that critical system components have another identical component with the same data, that can take over in case of failure.
- **Monitoring**—identifying problems in production systems that may disrupt or degrade service.
- **Failover**—the ability to switch from an active system component to a redundant component in case of failure, imminent failure, degraded performance or functionality.
- **Fallback**—the ability to switch back from a redundant component to the primary active component, when it has recovered from failure.

# High Availability

---

What is difference between availability and reliability a cloud?

Availability in cloud computing ensures that products, services and tools are available to customers at any given point of time and is measured by downtimes of overall cloud services.

Reliability, however, measures the probability that the cloud services are delivered exactly what they are meant for

## **High Availability Metrics: RTO and RPO**

The two metrics normally used to assess HA as well as disaster recovery (DR) are the recovery time objective (RTO) and the recovery point objective (RPO).

- RTO is the maximum tolerable duration of any outage. Online transaction processing applications generally have the lowest RTOs, and those that are essential often have an RTO of only a few minutes.
- RPO is the maximum amount of data loss that can be tolerated when a failure happens. For HA, RPO is often zero to specify there should be zero data loss under all failure scenarios.

# High Availability

---

## High Availability Requirements in Cloud

---

- **What is meant by High Availability ?**
  - Defined by Service Level Agreement (SLA):
  - HA goal is to meet SLA requirement
  - Balance between the availability and implementation cost
  - SLA: for example, 99.95%, annual 4 hrs 22 minutes downtime  
Downtime window: first Saturday: 8pm-10pm every quarter
- **Cases impacting system availability:**
  - Service outage by unplanned downtime:  
hardware or software failure, human error
  - Service disruption by planned downtime:  
hardware/software upgrade, patching and migration from old system to new system
  - Service performance degrade: violate performance SLA  
for example, 99% transactions finished in a 2 seconds window

# High Availability

---

Availability (also known as service availability) is both a commonly used metric to quantitatively measure resiliency, as well as a target resiliency objective.

Availability is the percentage of time that a workload is available for use.

Available for use means that it performs its agreed function successfully when required. This percentage is calculated over a period of time, such as a month, year, or trailing three years. Applying the strictest possible interpretation, availability is reduced anytime that the application isn't operating normally, including both scheduled and unscheduled interruptions.

We define availability as follows:

- Availability is a percentage uptime (such as 99.9%) over a period of time (commonly a month or year) • Common short-hand refers only to the “number of nines”; for example, “five nines” translates to being 99.999% available
- Some customers choose to exclude scheduled service downtime (for example, planned maintenance) from the Total Time in the formula.

**However, this is not advised, as your users will likely want to use your service during these times.**

**Availability=Availability of Use time/ Total Time**

# High Availability

---

Calculating availability with hard dependencies.

Many systems have hard dependencies on other systems, where an interruption in a dependent system directly translates to an interruption of the.

- . Where such hard dependencies occur, the invoking system's availability is the product of the dependent systems' availabilities.

For example, if you have a system designed for 99.99% availability that has a hard dependency on two other independent systems that each are designed for 99.99% availability, the workload can theoretically achieve 99.97%

availability:  $\text{Availinvok} \times \text{Availdep1} \times \text{Availdep2}$

$$= \text{Availworkload } 99.99\% \times 99.99\% \times 99.99\% = 99.97\%$$

It's therefore important to understand your dependencies and their availability design goal

# High Availability Calculation

---

Calculating availability with redundant components.

When a system involves the use of independent, redundant components (for example, redundant resources in different Availability Zones),

the theoretical availability is computed as 100% minus the product of the component failure rates.

For example, if a system makes use of two independent components, each with an availability of 99.9%, the effective availability of this dependency is 99.9999%:

$$\text{Avail}_{\text{effective}} = \text{Avail}_{\text{MAX}} - ((100\% - \text{Avail}_{\text{dependency}}) \times (100\% - \text{Avail}_{\text{dependency}}))$$
$$99.9999\% = 100\% - (0.1\% \times 0.1\%)$$

Shortcut calculation: If the availabilities of all components in your calculation consist solely of the digit nine, then you can sum the count of the number of nines digits to get your answer. In the above example two redundant, independent components with three nines availability results in six nines.

Calculating dependency availability.

(for example, a component where the manufacturer does not publish availability information), one way to estimate is to determine the Mean Time Between Failure (MTBF) and Mean Time to Recover (MTTR). An availability estimate can be established by: For example, if the MTBF is 150 days and the MTTR is 1 hour, the availability estimate is 99.97%

# AWS High Availability

---

## AWS high availability architecture

AWS has a global infrastructure to provide high availability for cloud workloads. The key components of this architecture include:

- **Regions**—21 geographical zones each containing at least three availability zones.
- **Availability zones**—66 global zones, which are self-sufficient data centers with redundant power, networking and cooling. Deploying across several AZs can protect your applications and provide you with resiliency in case failures occur.
- **Compliance and data residency**—Amazon provides full control over AWS regions to help you comply with data sovereignty requirements.

# AWS High Availability

---

## Advantages of Using AWS High Availability for Web Applications

AWS high availability for web applications provides you with the following benefits:

- A completely secured network that uses a Web Application Firewall (WAF) to prevent common web exploits.
- AWS HA has provisions like Business Continuity (BC) and Disaster Recovery (DR) technologies to help businesses resume operations with minimal disruption.
- For cases where instant hardware failure may arise or are about to arise, AWS Auto Scaling automatically detects this and launches a new instance.
- AWS HA provides metrics on the cloud to closely monitor the application based on the number of users using the application or the memory consumed by the particular instance.
- The deployment of new features or updates may be done without causing any problems for present users.

# AWS High Availability

---

## AWS High Availability for Storage Services

Here is a brief summary of the high availability capabilities Amazon offers for other popular storage services:

### Amazon S3

S3 guarantees 99.99999999% (twelve 9's) durability, by redundantly storing objects on multiple devices across a minimum of three AZs in an Amazon S3 Region.

### Amazon EFS

EFS guarantees 99.9% availability, otherwise between 10-100% of the service fee is discounted. Every file system object is redundantly stored across multiple AZs.

### Amazon EBS

[EBS volumes](#) are created in a specific AZ. You can make a volume available in another AZ and it can then be attached to other instances in that same Availability Zone. To make a volume available outside the AZ, or to [create redundancy](#), you can create a snapshot and restore it in another AZ within the same region. You can also copy snapshots to other AWS regions, to create redundancy across Amazon data centers.

# AWS High Availability

---

## AWS Regions and High Availability Zones

- Amazon hosts its web services across multiple locations, with each AWS location consisting of multiple availability zones and availability ranging from 99.9% to 99.999%.
- Each AWS Region runs in complete autonomy. This ensures the greatest level of fault tolerance and stability for user and application workloads.
- All AWS Availability Zones (AZs) are configured to operate in such a way that they are able to provide inexpensive, low latency network connectivity to other Availability Zones in the same region as well. These are connected to multiple Internet Service Providers (ISPs) and different power grids.
- Your application(s) can be safeguarded against failure in a single data center by deploying EC2 instances in various Availability Zones.
- It is important to run independent application stacks in more than one Availability Zone, either in the same region or in another region, so that if one zone fails, the application in the other zone can continue to run.

# High Availability

---

The following elements help you implement highly available systems:

- **Redundancy**—ensuring that critical system components have another identical component with the same data, that can take over in case of failure.
- **Monitoring**—identifying problems in production systems that may disrupt or degrade service.
- **Failover**—the ability to switch from an active system component to a redundant component in case of failure, imminent failure, degraded performance or functionality.
- **Fallback**—the ability to switch back from a redundant component to the primary active component, when it has recovered from failure.

AWS helps you achieve high availability for cloud workloads, across three different dimensions:

- **Compute**—Amazon EC2 and other services that let you provision computing resources, provide high availability features such as load balancing, auto-scaling and provisioning across Amazon Availability Zones (AZ), representing isolated parts of an Amazon data center.
- **SQL databases**—Amazon RDS and other managed SQL databases provide options for automatically deploying databases with a standby replica in a different AZ.
- **Storage services**—Amazon storage services, such as S3, EFS and EBS, provide built-in high availability options. S3 and EFS automatically store data across different AZs, while EBS enables deployment of snapshots to different AZs.

# High Availability

---

## AWS Services Used to Achieve High Availability

AWS delivers high availability through a scalable, load-balanced cluster or an active-standby pair, among other approaches. The majority of Amazon Web Services are designed to be fault-tolerant and have high availability. The following list includes some of them:

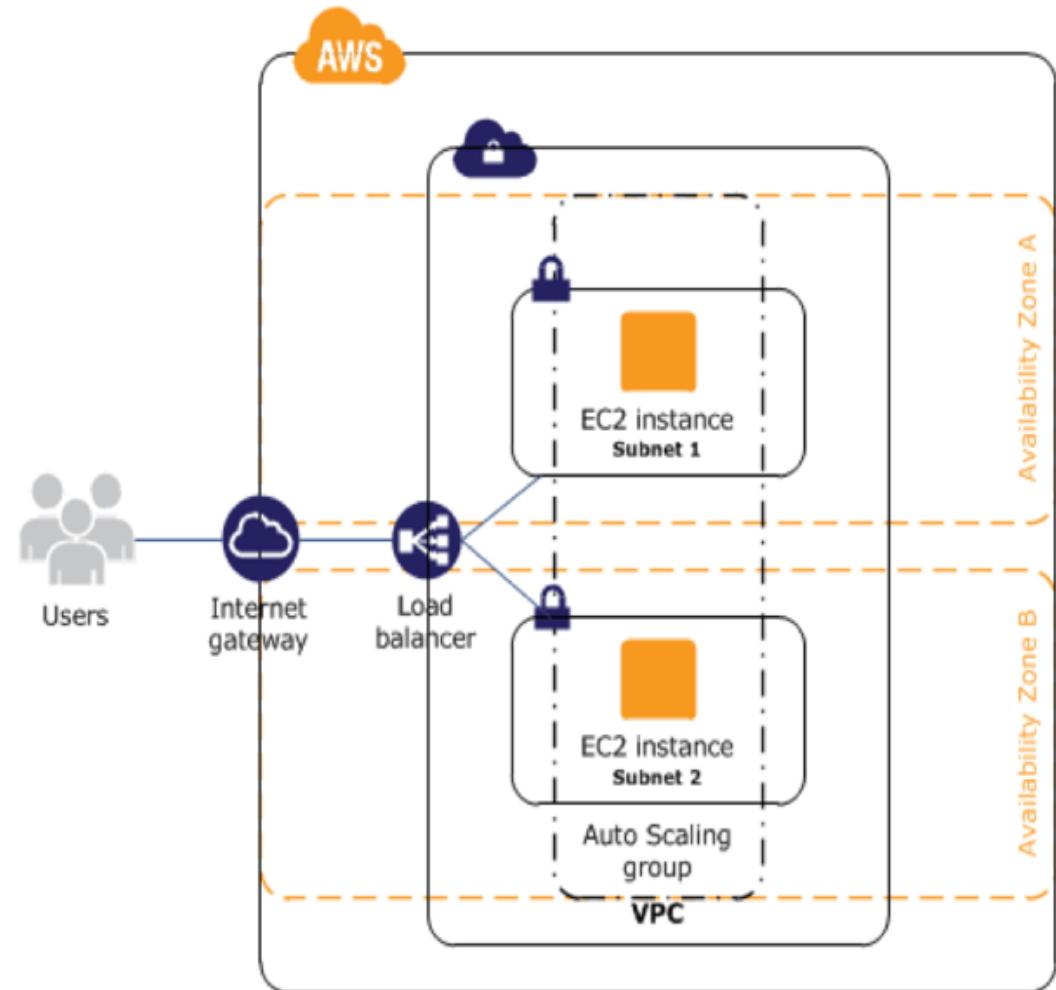
- Amazon S3
- Amazon Relational Database (RDS)
- Amazon Simple Queue Service (SQS)
- Elastic Load Balancing (ELB)
- Amazon Simple Notification Service (SNS)
- Amazon Virtual Private Cloud (VPC)
- Amazon Machine Engine (AMI)

# High Availability

## AWS High Availability for EC2 Instances

If you are running instances on Amazon EC2, Amazon provides several built-in capabilities to achieve high availability:

- **Elastic Load Balancing**—you can launch several EC2 instances and distribute traffic between them.
- **Availability Zones**—you can place instances in different AZs.
- **Auto Scaling**—use auto-scaling to detect when loads increase, and then dynamically add more instances.



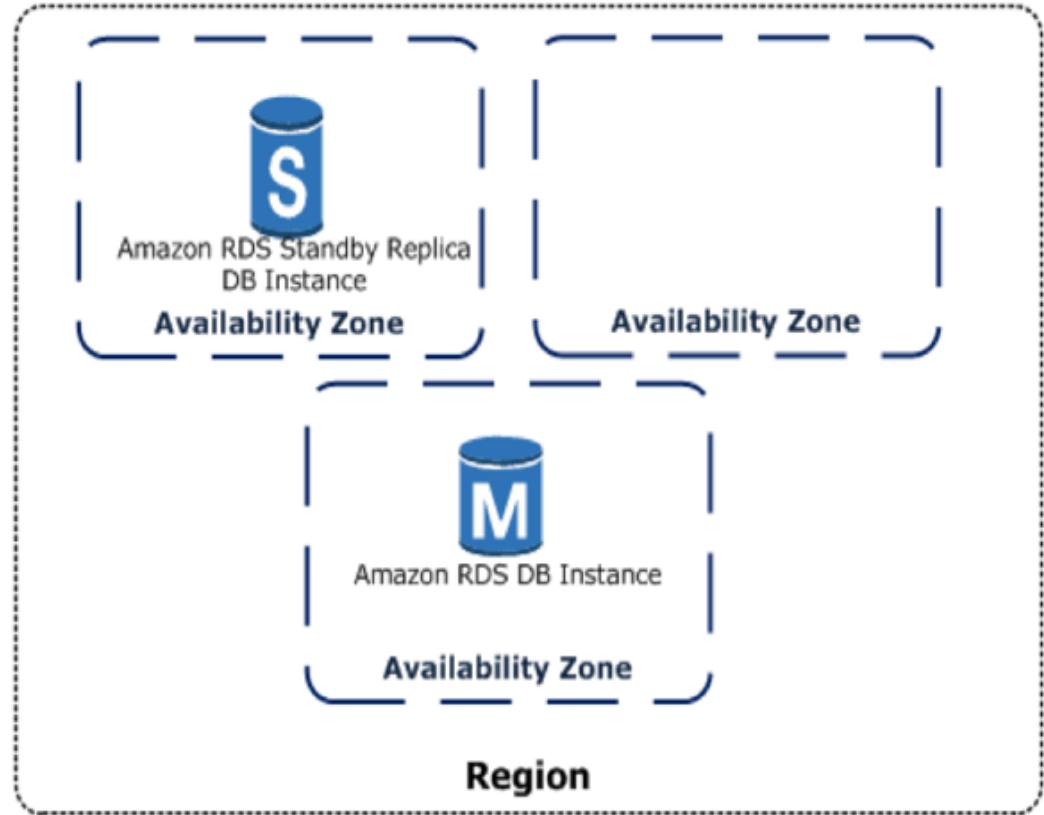
# High Availability

## AWS High Availability for SQL Databases on Amazon RDS

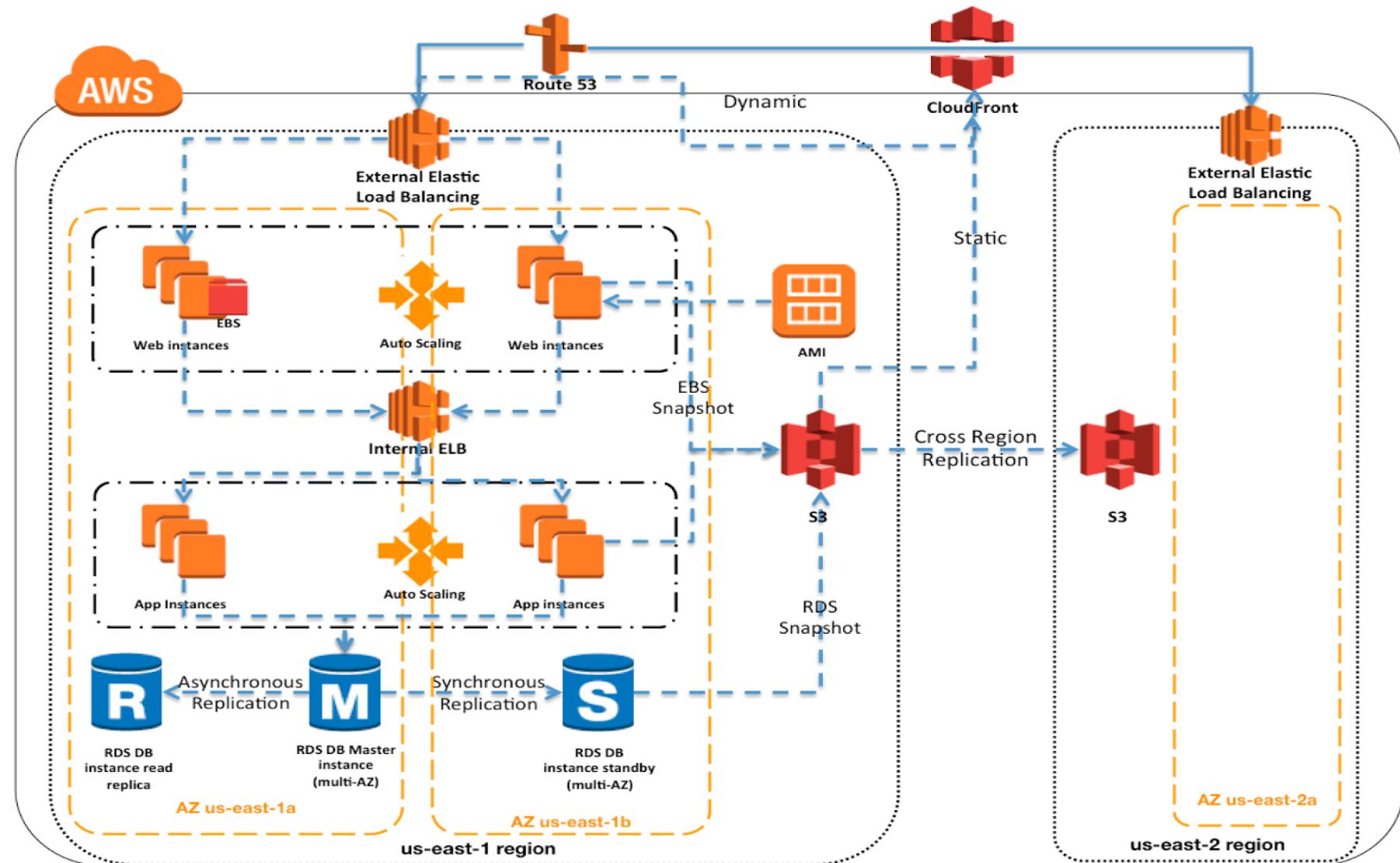
RDS provides high availability using Multi-Availability Zone (Multi-AZ) deployments. This means RDS automatically provisions a [synchronous replica](#) of the database in a different availability zone. When the main database instance goes down, users are redirected transparently to the other availability zone.

This provides two levels of redundancy:

- **In case the active database fails**, there is a standby replica ready to receive requests
- **In case of a disruption** in the AZ your main database instance is running in, there is failover to another AZ.



# High Availability



# What is SLA, SLO and SLI

---

**SLA**



**SERVICE LEVEL AGREEMENT**

the agreement you make  
with your clients or users

**SLOs**



**SERVICE LEVEL OBJECTIVES**

the objectives your team must  
hit to meet that agreement

**SLIs**



**SERVICE LEVEL INDICATORS**

the real numbers on  
your performance

# SLA, SLO and SLI Goal

---

## SLA

Promise \_\_\_\_\_

Promise \_\_\_\_\_

Promise \_\_\_\_\_

## SLOs

Goal \_\_\_\_\_

Goal \_\_\_\_\_

Goal \_\_\_\_\_

## SLIs

How did  
we do? \_\_\_\_\_

How did  
we do? \_\_\_\_\_

How did  
we do? \_\_\_\_\_

## What Is An SLA?

---

SLA stands for the “service-level agreement.” It is an agreement between a party that offers some service(s) and users of those service(s). The contract includes the list of services and highlights the quality standards that the provider should follow to guarantee customer satisfaction. The contract also recalls the ways to redress gaps and problems (e.g., using service credits).

### **What Should Service Level Agreement Include:**

- Detailed service overview
- Speed of service delivery
- Plan for performance monitoring
- Description of the reporting procedure
- List of penalties that will be applied in case of agreement violations
- Constraints

## What is SLO and SLI

---

An SLO (service level objective) is an agreement within an SLA about a specific metric like uptime or response time. So, if the SLA is the formal agreement between you and your customer, SLOs are the individual promises you're making to that customer. SLOs are what set customer expectations and tell IT and DevOps teams.

An SLI (service level indicator) measures compliance with an SLO (service level objective). So, for example, if your SLA specifies that your systems will be available 99.95% of the time, your SLO is likely 99.95% uptime and your SLI is the actual measurement of your uptime. Maybe it's 99.96%. Maybe 99.99%. To stay in compliance with your SLA, the SLI will need to meet or exceed the promises made in that document.

## What Is An SLA?

---

### What Is An SLA Model?

To understand that, one may need to know the performance metrics that are usually involved:

- Availability & uptime percentage
- Various performance benchmarks
- Response time (often associated with the work of customer service)
- Problem resolution time
- Usage statistics that will be provided

## SLA Types

---

SLA is usually divided into 3 categories. They are:

- Customer-based SLA

This type of SLA is intended for individual customers and includes all services they request. The document covers details about service quality to let customers know what level of service delivery they should expect. A good example could be telecommunication companies. Their services contain messaging, Internet connection, and voice calls. However, all of them fall under just one agreement.

- Service-oriented SLA

This document presents a single identical service for all clients. It is based on a single set of standards which makes this type of SLA the most convenient one. For instance, users sign the service level agreement regarding information technologies helpdesk. That means that the same service is valid for all users.

- Multi-level SLA

This agreement is based on requests from end-user companies. It's a customizable contract that makes it possible to play with various standards and conditions to make both sides benefit from the final version of the document. This type of SLA can be divided into subcategories.

1. *Corporate:* No frequent updates are needed for such an agreement. The points included in the contract usually remain unaltered. The document of this type is applicable to all customers of the company.

2. *Client:* Such an agreement covers all service aspects related to a certain category of customers. At the same time, the type of services is not considered.

## What makes a good SLA?

---

A well thought out content. Here are six components necessary for a good agreement:

- **Document overview**

This involves checking main points listed in the agreement: individuals involved, start and expiration dates and other details.

- **Strategic goals**

Description of the agreement purpose and objectives.

- **Shareholders**

A list of all parties and individuals involved in the contract.

- **Regular review and necessary changes**

This section demonstrates a requirement for systematic review and conditions upon which modifications can be made.

- **Service agreement**

This section covers customer and provider requirements, service scope and service assumptions.

- **Service management**

Includes information about service delivery and availability.

service delivery and availability.

Level of Availability	Percent of Uptime	Downtime Per Year	Downtime Per Day
1-9	90%	36.5 DAYS 	2.4 HRS. 
2-9	99%	3.65 DAYS 	14 MINS. 
3-9	99.9%	8.76 HRS. 	86 SECS. 
4-9	99.99%	56.2 MINS. 	8.6 SECS. 
5-9	99.999%	5.25 MINS. 	.86 SECS. 
6-9	99.999%	31.56 SECS. 	8.6 MSECS. 

## What professional SLA management services

---

What professional SLA management services should include:

- Setting realistic conditions that a service provider can ensure;
- Meeting the needs and requirements of the clients;
- Establishing the right metrics for evaluating the performance of the services;
- Ensuring compliance with the terms and conditions agreed with the clients;
- Avoiding any violations of SLA terms and conditions.

An SLA is a preventive means that allows establishing a transparent relationship between both parties involved and increases confidence in the cooperation. Such a document is fundamental to a successful collaboration between a client and a service provider.

## Types Of SLA Penalties

---

A natural reply to any kind of violation is penalty. An SLA penalty depends on the industry and business. Let's take a look at the two most common SLA penalty types.

### 1. Financial penalty

This kind of penalty requires a vendor to pay the customer a compensation of damages equal to the one written in the agreement. The amount will depend on the extent of a violation and damage and may not be a full reimbursement of what a customer paid for the ecommerce service or [ecommerce support](#).

### 2. Service credit

In this case a service provider will reimburse the customer in the form of service credits which a client can use for future projects. In other words, a vendor will have to provide a customer with free services for a specific time period.

To avoid any confusion or misunderstanding between the two parties in case of SLA violation such penalties must be clearly articulated in the agreement. Otherwise, they won't be legitimate. The conditions of paying off the compensation should be stated explicitly and in details without leaving any room for discussion.

## What are the differences between SLA and KPI?

It is common for some people to mix SLA and [KPI](#), but they are two very different concepts. In short, companies build an SLA to ensure complying services until the end of the contract. KPIs, on the other hand, are performance indicators aimed at measuring actions already taken.

Thus, **the SLA anticipates what may happen** and tries to ensure that everything goes well, while **the KPI brings predefined indicators** to measure the results of the actions put into practice.

# What are the best SLA practices?

---

## **Set an average response time for problem-solving**

When a client calls your business because of a problem, the teams in charge need to **perform a quick response time**. After all, the longer the problem persists, the bigger the delays and damages to work.

when defining your SLA, consider potential problems and the average time it takes to solve them. This way, everyone will have the right expectation about response time.

## **Use neutral language between the parties**

It is important to use terms that are clear and simple for everyone to understand. Even if you are able to understand what your IT manager says, particular words may be difficult to understand for your clients and even colleagues at the company. In this context, neutral language is key to ease communication.

## **Simplify the creation of SLAs**

Instead of creating long and complex SLAs, set simple and shorter versions, so you can analyze their flow and the customer can understand them. This action makes the workflow more appropriate to optimize each task your team must do.

## **Define which tickets have higher priority**

Imagine that, during a work shift, one of the headsets stops working in an important call and the [CRM](#) access goes down. Your IT manager will receive two tickets at the same time. So which one to prioritize?

Answering this question requires you to define the different ticket priorities in advance, as the weight can be higher or lower depending on the circumstance.

In the example above, access to the CRM would be a higher priority because it affects everyone's work, while the damaged headset concerns only one person. However, if the call is to talk to a customer who is about to churn, that priority might **be higher**.

## What are the best SLA practices?

---

### Determine days and times for SLA tasks

A customer may get frustrated when they contact you and do not get a response. But what if they made contact on a Saturday midnight instead of Monday through Friday business hours?

If your company operates Monday through Friday during business hours, it has to be clear in the SLA. This way, customers are aware that your team will only perform tasks and requests during the given days and times.

On the other hand, if the service is outsourced and provides 24/7 support, it is still essential to insert in the contract what types of services are provided during and outside business hours. This way, customers and everyone inside the company know what to expect.

# Key aspects of SLA

---

cloud service provider. Service level agreements usually specify certain parameters, which are mentioned below:

- Availability of the Service (uptime)
- Latency or the response time
- Service components reliability
- Each party accountability
- Warranties

# Types of SLA

---

What are the three types of SLAs? There are three basic types of SLAs:  
**customer, internal and multilevel service-level agreements.**

A customer service-level agreement is between a service provider and its external customers

## **The main elements of a good SLA.**

- Overall objectives. The SLA should set out the overall objectives for the services to be provided
- Description of the Services. The SLA should include a detailed description of the services
- Performance Standards
- Compensation/Service Credits.
- Critical Failure
- .

## The importance of a cloud SLA

Service-level agreements are fundamental as more organizations rely on external providers for their critical systems, applications and data.

A cloud SLA ensures cloud providers meet certain enterprise-level requirements and provide customers with a clearly defined set of deliverables. It also describes financial penalties, such as credits for service time, if the provider fails to live up to the guaranteed terms.

Cloud service-level agreements may be more detailed to cover governance, security specifications, compliance, and performance and uptime statistics. They should address security and encryption practices for data protection and data privacy, disaster recovery expectations, data location, as well as data access and portability.

---

# Key aspects of SLA

---

Typically, the **service-level objectives** (SLOs) for these applications were response time and throughput of the application end-user requests.

The capacity buildup was to cater to the estimated peak load experienced by the application. The activity of determining the number of servers and their capacity that could satisfactorily serve the application end-user requests at peak loads is called capacity planning

# Key aspects of SLA

What is an SLI?

An SLI, or Service Level Indicator, measures how well a company actually meets the SLO promises that it sets within SLAs.

For instance, if you promise an SLO of 99.9 percent uptime, and you achieve 99.95 percent uptime, then your SLI would be 99.95 percent (and you'd be exceeding your SLO, which is good).

Tracking SLIs is important for two main reasons. The most obvious is that demonstrating SLIs to customers allows companies to show that they are meeting the terms of an SLA.

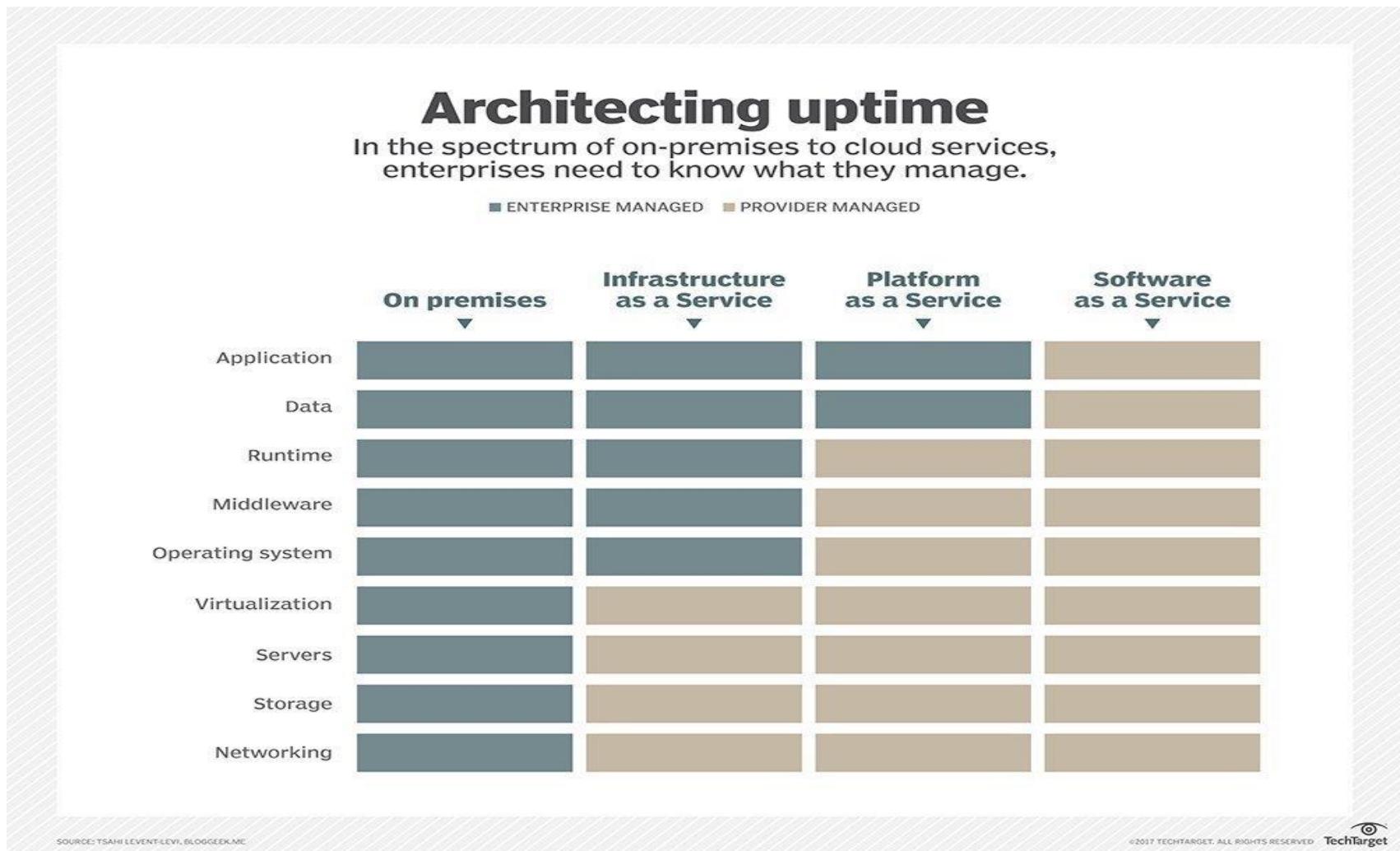
Second, by tracking SLIs on a continuous basis, vendors can detect when they are falling short of meeting SLO promises, and they can take measures to address the problem before it turns into an SLA violation. Given that SLOs are often made on the basis of meeting certain goals over a period of time, early detection of SLI issues makes it possible to correct those issues before they persist long enough to trigger SLO non-compliance.

Similarities and differences between SLAs, SLOs and SLIs

SLAs, SLOs and SLIs share one major thing in common: They are all part of the formal process that businesses use to set and track reliability, performance and availability goals. By extension, they are central to the [work performed by SREs](#), whose main job is to help businesses meet the goals they set within these categories.

However, once you dive into the details, SAs, SLOs and SLIs are clearly different types of entities:

- An SLA is a contract.
- An SLO is a specific goal that is defined in a contract.
- An SLI measures the extent to which teams comply with the SLO promises they make in SLA contracts.



# Key aspects of SLI

An SLI, or Service Level Indicator, measures how well a company actually meets the SLO promises that it sets within SLAs.

For instance, if you promise an SLO of 99.9 percent uptime, and you achieve 99.95 percent uptime, then your SLI would be 99.95 percent (and you'd be exceeding your SLO, which is good).

Tracking SLIs is important for two main reasons. The most obvious is that demonstrating SLIs to customers allows companies to show that they are meeting the terms of an SLA.

Second, by tracking SLIs on a continuous basis, vendors can detect when they are falling short of meeting SLO promises, and they can take measures to address the problem before it turns into an SLA violation. For example, if an incident disrupts your application's availability and you can resolve the incident quickly with the help of an incident management platform, you can fix the problem before it results in enough downtime to cause you to fall short of your SLO promises.

Given that SLOs are often made on the basis of meeting certain goals over a period of time, early detection of SLI issues makes it possible to correct those issues before they persist long enough to trigger SLO non-compliance.

# Benefits of service level management

---

What is service level management?

Service level management means ensuring that all processes and operational agreements for the level of services provided to customers are appropriate.

It includes monitoring and reporting on service levels, setting and adjusting SLOs, determining SLIs, making sure you are meeting SLAs, and holding customer reviews.

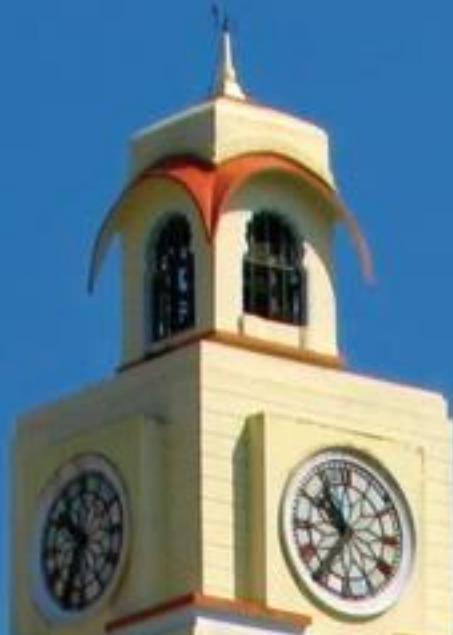
The central focus really is the shared meaning of “availability” across teams, in your SLOs, also captured in the SLAs with your customers. To make sure your business is meeting or exceeding these service level agreements, it’s important for cross-functional teams to manage internal SLOs.

# Benefits of service level management



Good practices for SLIs, SLOs, and SLAs, and a platform for your service level management, bring these benefits:

- **Easy setup:** Automatically establish a baseline of performance and reliability for any service with a one-click setup and recommendations and customizations provided in a simple, guided flow.
- **Define reliability across teams:** Avoid arduous alignment processes with SLO and SLI recommendations that help you determine service boundaries. Set reliability benchmarks automatically based on recent performance metrics in any entity.
- **Iterate and improve:** With full-stack context and teams have insight into how specific nodes or services impact system reliability and can quickly take control over their performance. Custom views for both service owners and business leaders drive operational efficiency and lead to better reporting, alerting, and incident management processes.
- **Standardize reliability:** Cross-organizational teams have a unified, transparent view of service reliability, and can better comply with customer-facing SLAs. SLO compliance metrics and error budgets give organizations a way to report on reliability and implement changes across applications, infrastructure, and teams in a cohesive fashion.



# Cloud Computing

## SEWP ZG527

**BITS** Pilani

# AGENDA



- Big Data Growth Drivers
- What is Big Data?
- Hadoop Introduction
- Hadoop Master/Slave Architecture
- Hadoop Core Components
- HDFS Data Blocks
- HDFS Read/Write Mechanism
- What is MapReduce
- MapReduce Program
- MapReduce Job Workflow
- Hadoop Ecosystem



## **Big Data Growth Drivers**

# Big Data EveryWhere!

- Lots of data is being collected and warehoused
  - Web data, e-commerce
  - purchases at department/grocery stores
  - Bank/Credit Card transactions
  - Social Network



# How Much Data

- Google processes 20 PB a day
- Wayback Machine has 3 PB + 100 TB/month
- Facebook has 2.5 PB of user data + 15 TB/day
- eBay has 6.5 PB of user data + 50 TB/day
- CERN's Large Hydron Collider (LHC) generates 15 PB a year
- 1.7MB of data is generated every second by an individual in 2020
- Humans produce 2.5 quintillion bytes of data daily
- By 2025, humans shall generate 463 exabytes of data Each day:
- 306.4 billion emails are sent
- 95 million videos and photos are shared on Instagram
- 5 million Tweets are made.
-

## Type of Data

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), ...
- Streaming Data
  - You can only scan the data once

# What to do with these data?

- Aggregation and Statistics
  - Data warehouse and OLAP
- Indexing, Searching, and Querying
  - Keyword based search
  - Pattern matching (XML/RDF)
- Knowledge discovery
  - Data Mining
  - Statistical Modeling



What is Big Data?

# Introduction to Big Data

- 'Big Data' is similar to 'small data', but bigger in size but having data bigger it requires different approaches: –
- Techniques, tools and architecture with an aim to solve new problems or old problems in a better way
- Big Data generates value from the storage and processing of very large quantities of
- digital information that cannot be analyzed with traditional techniques
- Big Data may well be the Next Big Thing in the IT world.
- Big data burst upon the scene in the first decade of the 21<sup>st</sup> century.
- The first organizations to embrace it were online and startup firms.
- Firms like Google, eBay, LinkedIn, and Facebook were built around big data from the beginning.
- Like many new information technologies, big data can bring about dramatic cost reductions, substantial improvements in the time required to perform a computing task, or new product and service offerings.

# What is Big Data?

## Volume



Processing increasing huge data sets

## Variety



Processing different types of data

## Velocity



Data is being generated at an alarming rate

## Value



Finding correct meaning out of the data

## Veracity

Min	Max	Mean	SD
4.3	?	5.84	0.83
2.0	4.4	3.05	50000000
15000	7.9	1.20	0.43
0.1	2.5	?	0.76

Uncertainty and inconsistencies in data

large and complex

difficult

- A typical PC might have had 10 gigabytes of storage in 2000.
- Today, Facebook ingests 500 terabytes of new data every day.
- Boeing 737 will generate 240 terabytes of flight data during a single flight across the US.
- The smart phones, the data they create and consume;
- sensors embedded into everyday objects will soon result in billions of new, constantly-updated data feeds containing environmental, location, and other information, including video

- Clickstreams and ad impressions capture user behaviour at millions of events per second
- high-frequency stock trading algorithms reflect market changes within microseconds
- machine to machine processes exchange data between billions of devices
- infrastructure and sensors generate massive log data in realtime
- on-line gaming systems support millions of concurrent users, each producing multiple inputs per second.

- Big Data isn't just numbers, dates, and strings.
- Big Data is also geospatial data, 3D data, audio and video, and unstructured text, including log files and social media.
- Traditional database systems were designed to address smaller volumes of structured data, fewer updates or a predictable, consistent data structure.
- Big Data analysis includes different types of data

Choosing the correct data stores based on your data characteristics

- Analyzing your data characteristics
- Selecting data sources for analysis
- Moving code to data
- Implementing polyglot data store solutions
- Aligning business goals to the appropriate data store
- Eliminating redundant data
- **Overview of Big Data stores**
- Data models: key value, graph, document, column-family
- Hadoop Distributed File System, Hbase and Hive

- Integrating disparate data stores
- Mapping data to the programming framework
- Connecting and extracting data from storage
- Transforming data for processing
- Subdividing data in preparation for Hadoop MapReduce
- Employing Hadoop MapReduce
- Creating the components of Hadoop MapReduce jobs
- Distributing data processing across server farms
- Executing Hadoop MapReduce jobs
- Monitoring the progress of job flows



# Introduction to Apache Hadoop

Framework to Process Big Data

# Hadoop History

- **Dec 2004** – Google GFS paper published
- **July 2005** – Nutch uses MapReduce
- **Jan 2006** – Doug Cutting joins Yahoo!
- **Feb 2006** – Becomes Lucene subproject
- **Apr 2007** – Yahoo! on 1000-node cluster
- **Jan 2008** – An Apache Top Level Project
- **Feb 2008** – Yahoo! production search index

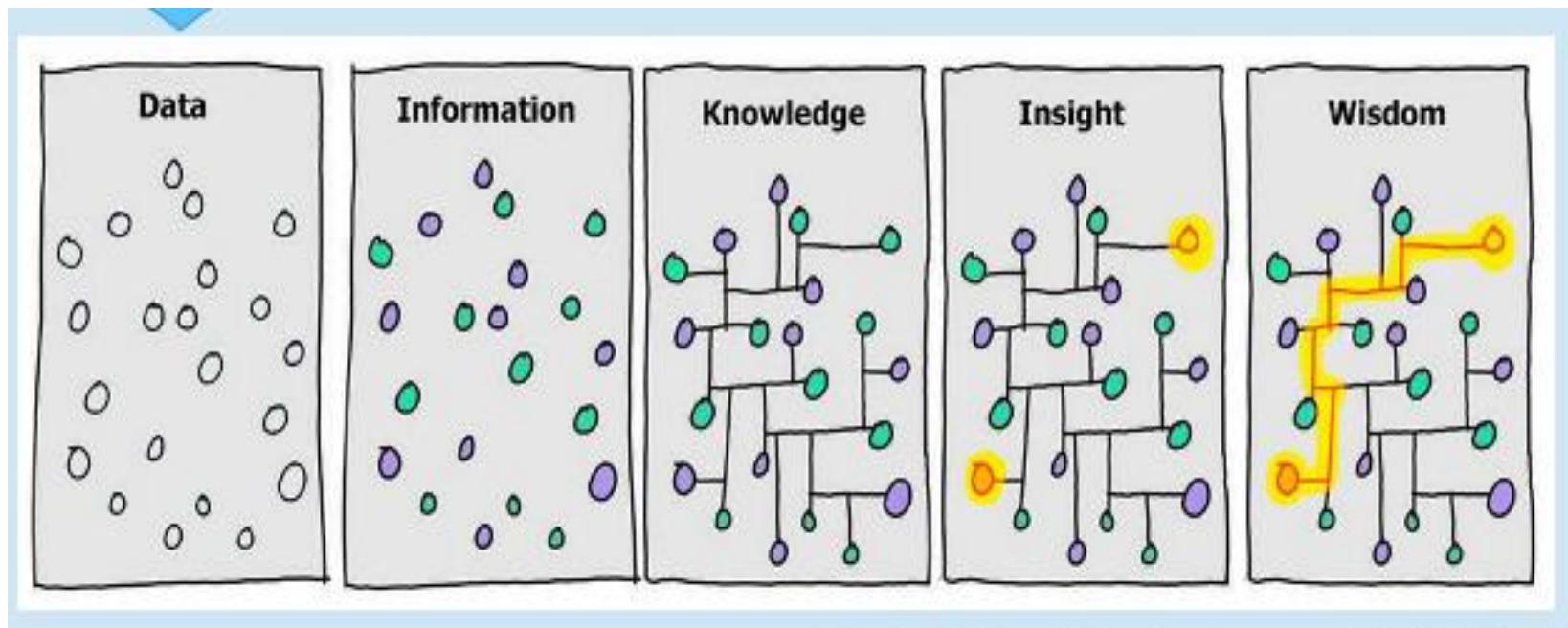
# What is Hadoop

- Hadoop is a simple software framework for *distributed processing* of *large datasets* across *huge clusters* of (commodity hardware) computers :
  - ***Large datasets*** ☐ Terabytes or petabytes of data
  - ***Large clusters*** ☐ Hundreds or thousands of nodes
- Open-source implementation for Google MapReduce
- Simple programming model : MapReduce
- Simple data model: flexible for any data

# HDFS File System Properties

- ***Large Space:*** An HDFS instance may consist of thousands of server machines for storage
- ***Replication:*** Each data block is replicated
- 
- ***Failure:*** Failure is norm rather than exception
- ***Fault Tolerance:*** Automated detection of faults and recovery

# Data Processing Stages



# Hadoop Vs RDBMS

## Summary : Hadoop vs. Typical DB

<b>Computing Model</b>	-Notion of transactions -Transaction is the unit of work -ACID properties, Concurrency control	-Notion of jobs -Job is the unit of work <b>-No concurrency control</b>
<b>Data Model</b>	-Structured data with known schema -Read/Write mode	-Any data format <b>-ReadOnly mode</b>
<b>Cost Model</b>	-Expensive servers	<b>-Cheap commodity machines</b>
<b>Fault Tolerance</b>	-Failures are rare -Recovery mechanisms	-Failures are common over thousands of machines -Simple fault tolerance
<b>Key Characteristics</b>	- Efficiency, Powerful, optimizations	- Scalability, flexibility, fault tolerance

# Comparison between Relational and Non-Relational DB

Architecture Drivers	Extended Relational	Non-Relational
Large data volume		
Self-service (ad-hoc reporting)		
Unstructured data processing		
High data model extensibility		
High data quality and consistency		
Extensive security		
Reliability and fault-tolerance		
Low latency (near-real time)		
Low cost		
Skills availability		

# Big Data Analytics

## Traditional Analytics (BI)

## vs Big Data Analytics

### Focus on

- Descriptive analytics
- Diagnosis analytics

- **Predictive analytics**
- **Data Science**

### Data Sets

- Limited data sets
- Cleansed data
- Simple models

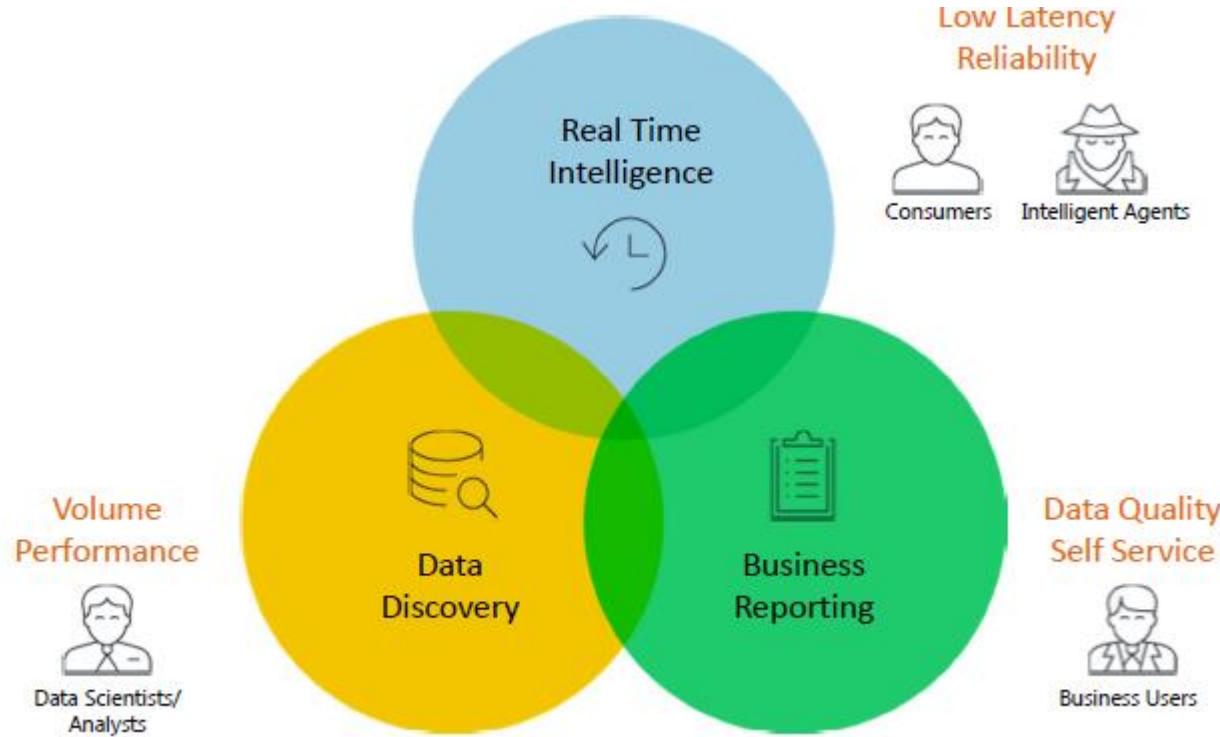
- Large scale data sets
- More types of data
- Raw data
- Complex data models

### Supports

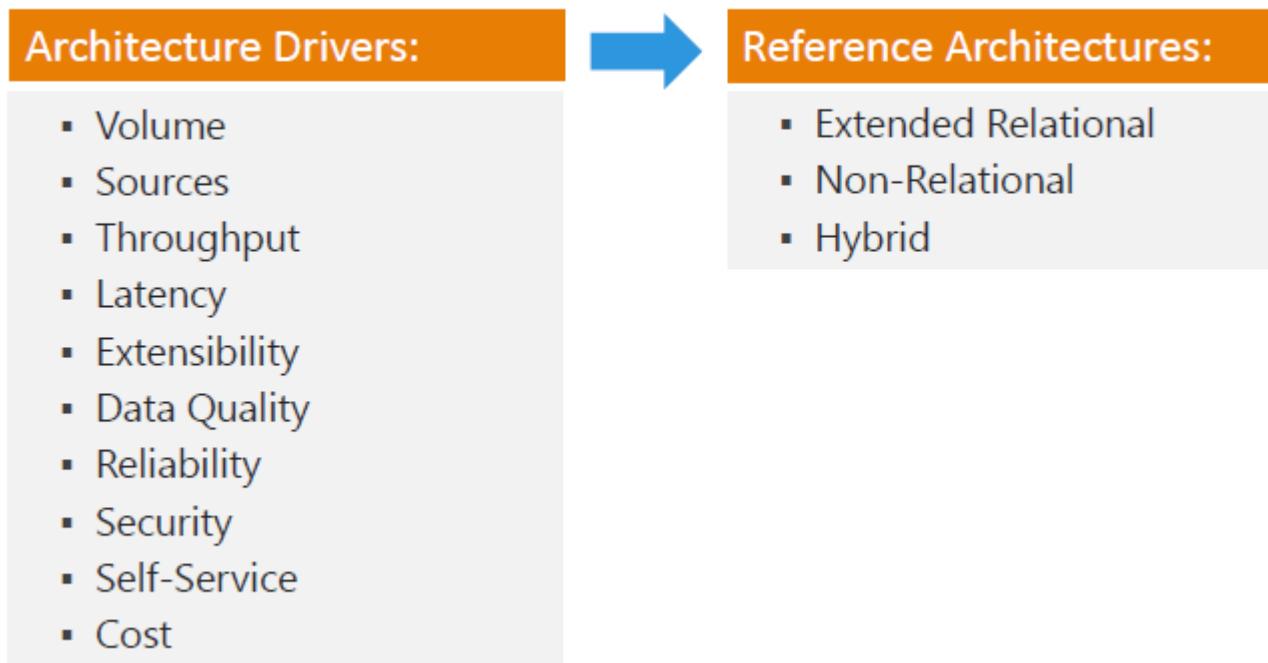
**Causation:** what happened, and why?

**Correlation:** new insight  
More accurate answers

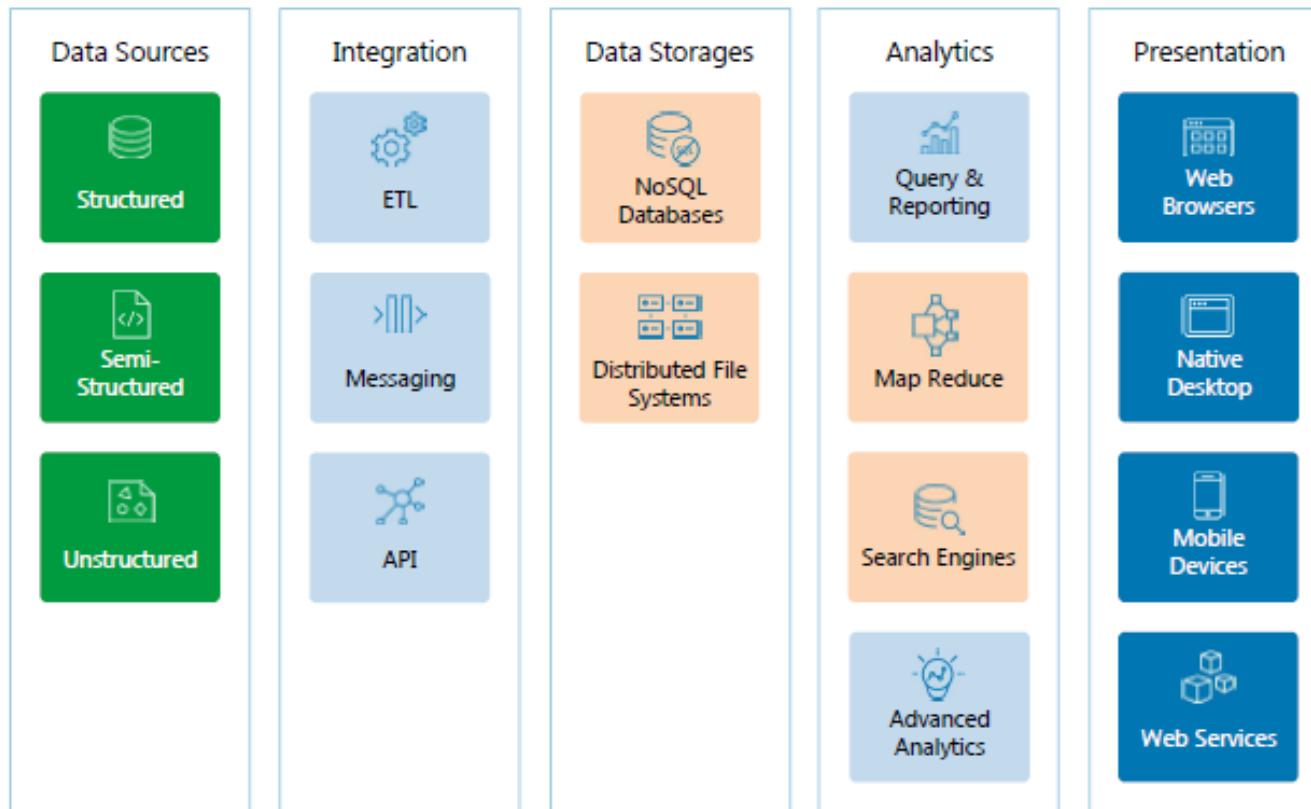
# Big Data use case



# Big Data Analytics Reference Architecture



# Non-Relational Reference Architecture



# Big Data Design Principles

- ❑ Understand data users and sources
- ❑ Discover architecture drivers
- ❑ Select proper reference architecture
- ❑ Do trade-off analysis, address cons
- ❑ Map reference architecture to technology stack
- ❑ Prototype, re-evaluate architecture
- ❑ Estimate implementation efforts
- ❑ Set up devops practices from the very beginning
- ❑ Advance in solution development through “small wins”
- ❑ Be ready for changes, big data technologies are evolving rapidly

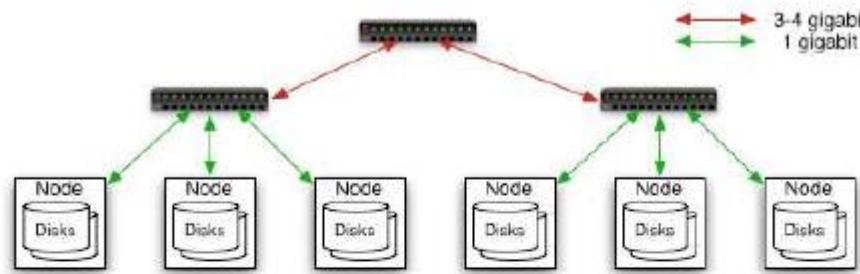
# Why Hadoop

- **Need to process huge datasets on large clusters of computers**
- **Very expensive to build reliability into each application.**
- **Nodes fail every day**
  - Failure is expected, rather than exceptional.
  - The number of nodes in a cluster is not constant.
- **Need common infrastructure**
  - Efficient, reliable, easy to use
  - Open Source, Apache License

# Who Uses Hadoop

- Amazon/A9
- Facebook
- Google
- IBM : Blue Cloud?
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

# Commodity Hardware



## Typically in 2 level architecture

- Nodes are commodity PCs
- 30-40 nodes/rack
- Uplink from rack is 3-4 gigabit
- Rack-internal is 1 gigabit

# Goal of HDFS

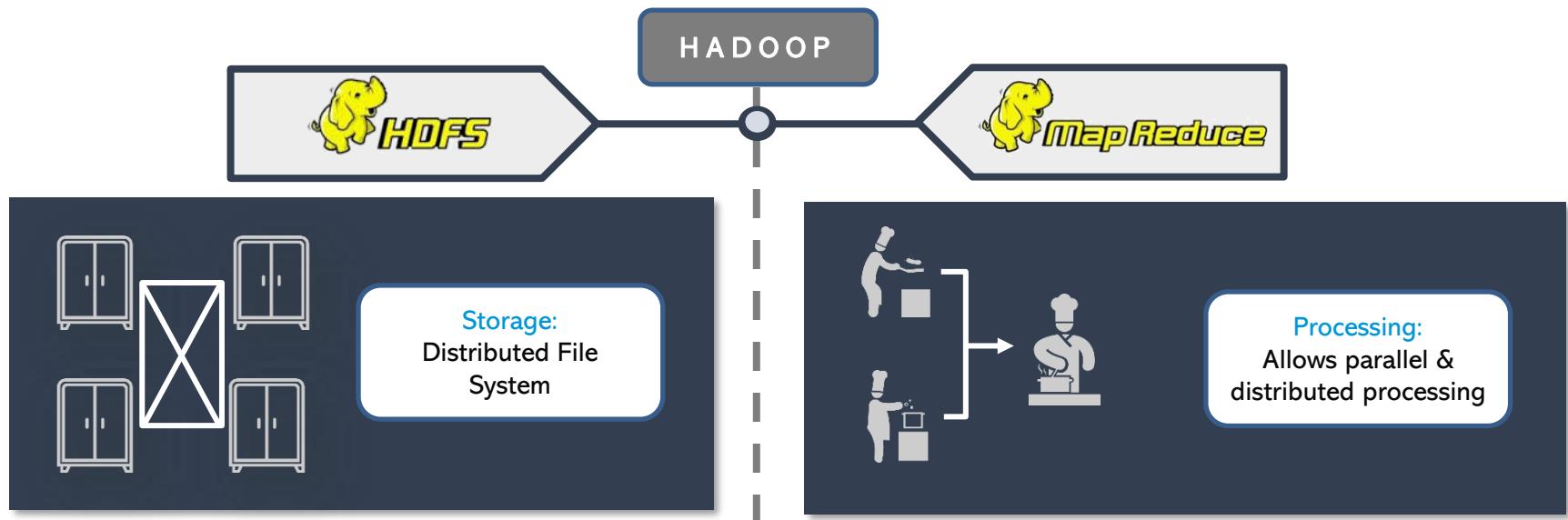
- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 PB
- **Assumes Commodity Hardware**
  - Files are replicated to handle hardware failure
  - Detect failures and recovers from them
- **Optimized for Batch Processing**
  - Data locations exposed so that computations can move to where data resides
  - Provides very high aggregate bandwidth

# Distributed File System

- **Single Namespace for entire cluster**
- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- **Files are broken up into blocks**
  - Typically 128 MB block size
  - Each block replicated on multiple DataNodes
- **Intelligent Client**
  - Client can find location of blocks
  - Client accesses data directly from DataNode

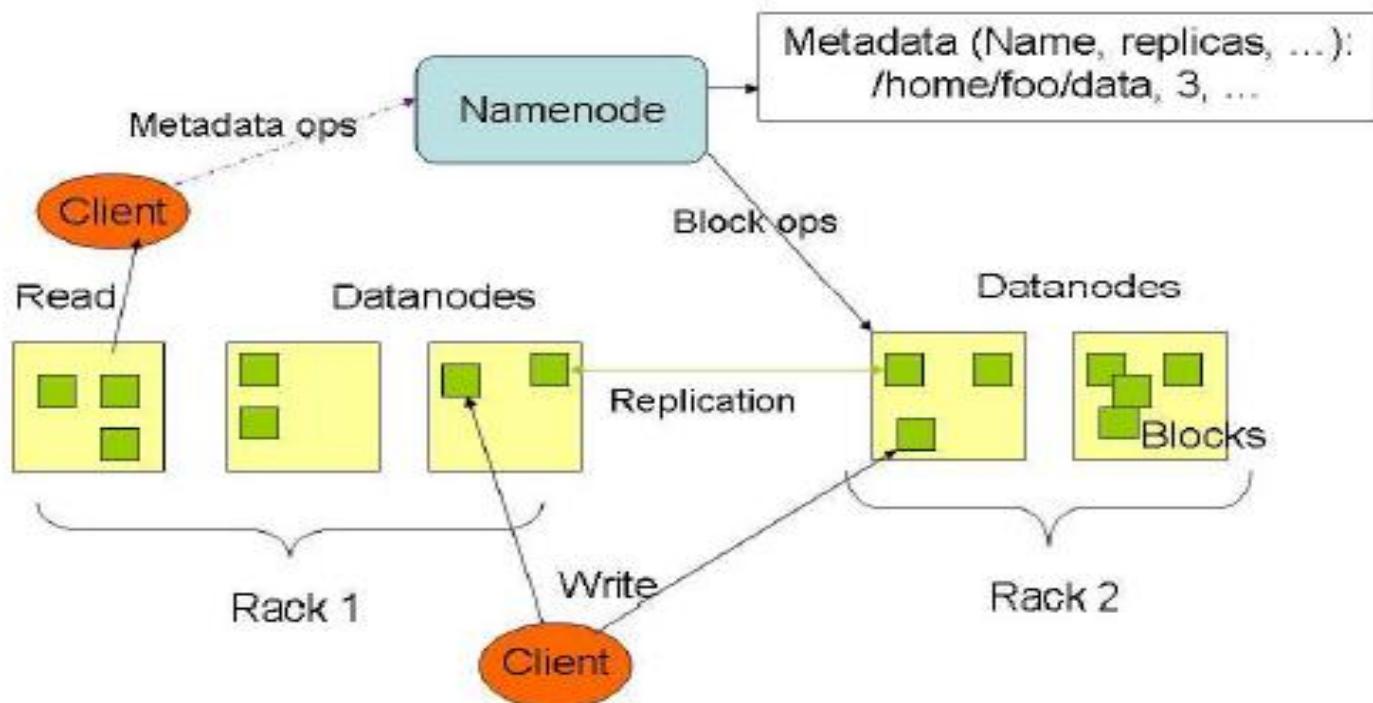
# Apache Hadoop Architecture

Hadoop is a **framework** that allows us to **store** and **process** large data sets in **parallel** and **distributed** fashion



# HDFS Architecture

## HDFS Architecture

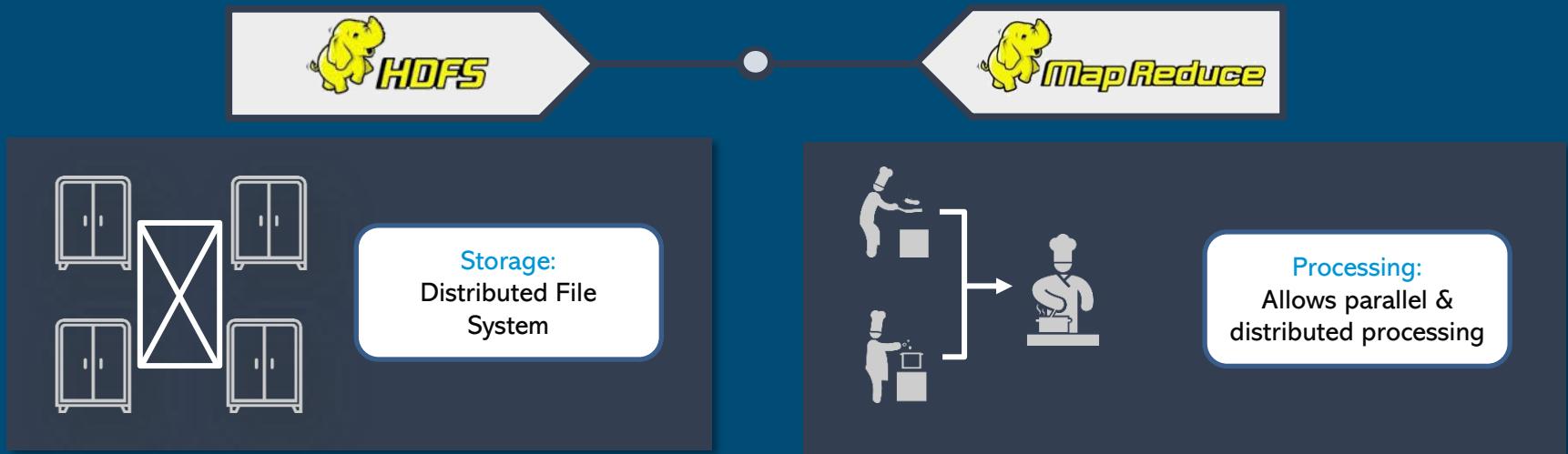




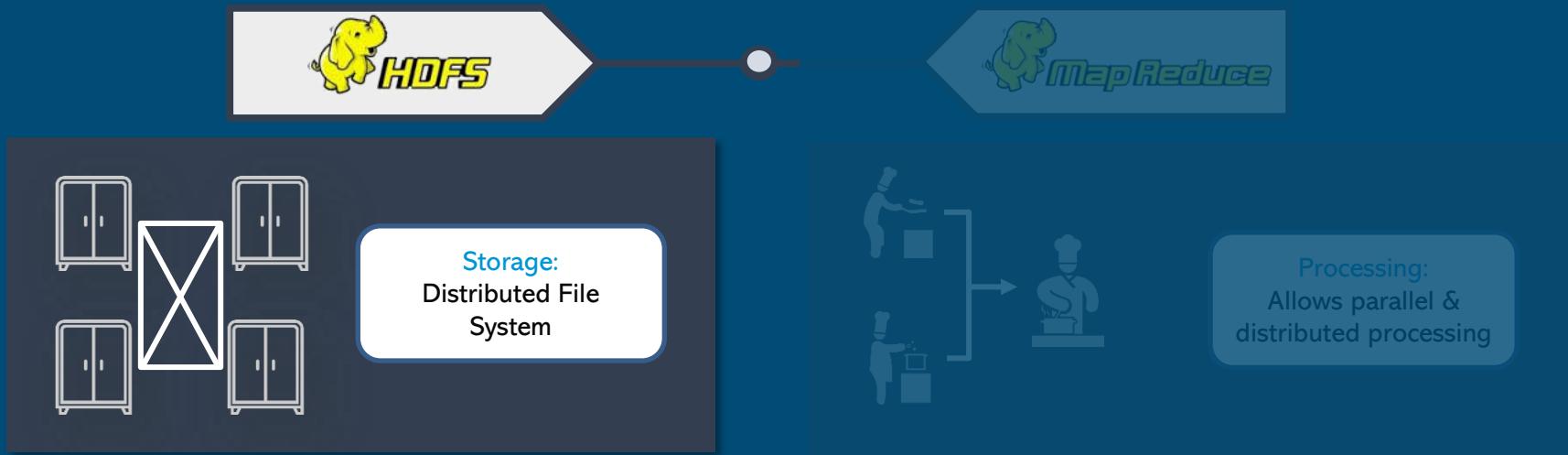
# Hadoop Core Components

Looking for Big Data Program? Call us at **IN: 8047474382/ US: +1 (415) 980-4399** or visit [www.edureka.co](http://www.edureka.co)

# Hadoop Core Components



# Hadoop Core Components

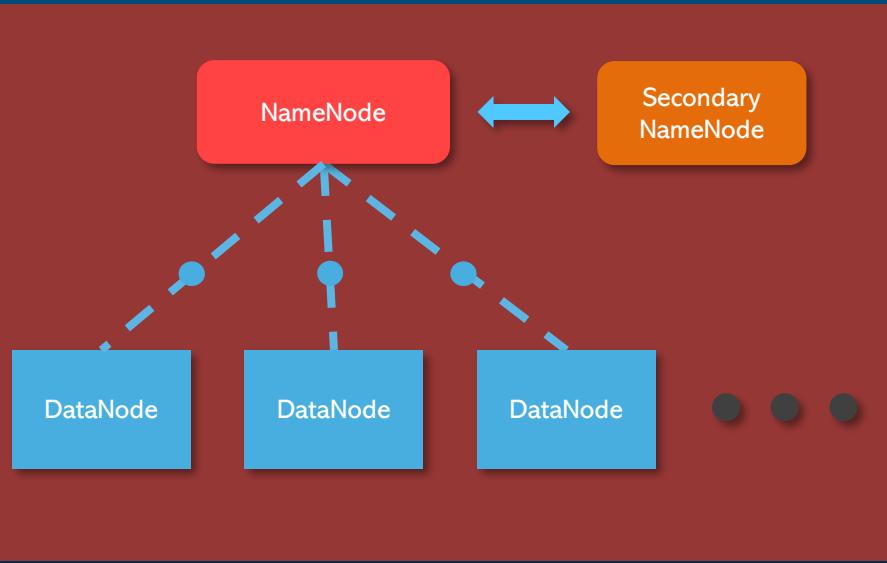




# HDFS Core Components

# HDFS Core Components





#### **NameNode:**

- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

#### **DataNode:**

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

# **NameNode & Data Node**

# Function of Name Node

- **Manages File System Namespace**
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides
- **Cluster Configuration Management**
- **Replication Engine for Blocks**

# Name Node Failures

- **A single point of failure**
- **Transaction Log stored in multiple directories**
  - A directory on the local file system
  - A directory on a remote file system (NFS/CIFS)
- **Need to develop a real HA solution**

# Secondary NameNode

- Copies FSIImage and Transaction Log from NameNode to a temporary directory
- Merges FSIImage and Transaction Log into a new FSIImage in temporary directory
- Uploads new FSIImage to the NameNode
  - Transaction Log on NameNode is purged

Hadoop FS-Image- FSIImage is a file stored on the OS filesystem that contains the complete directory structure (namespace) of the HDFS with details about the location of the data on the Data Blocks and which blocks are stored on which node. This file is used by the NameNode when it is started.

# Name Node Meta Data

- **Meta-data in Memory**
  - The entire metadata is in main memory
  - No demand paging of meta-data
- **Types of Metadata**
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g creation time, replication factor
- **A Transaction Log**
  - Records file creations, file deletions. etc

# Data Node

- **A Block Server**
  - Stores data in the local file system (e.g. ext3)
  - Stores meta-data of a block (e.g. CRC)
  - Serves data and meta-data to Clients
- **Block Report**
  - Periodically sends a report of all existing blocks to the NameNode
- **Facilitates Pipelining of Data**
  - Forwards data to other specified DataNodes

# Block Placement

- **Current Strategy**
  - One replica on local node
  - Second replica on a remote rack
  - Third replica on same remote rack
  - Additional replicas are randomly placed
- **Clients read from nearest replica**
- **Would like to make this policy pluggable**

# HeartBeats

- **DataNodes send heartbeat to the NameNode**
  - Once every 3 seconds
- **NameNode used heartbeats to detect DataNode failure**

# Replication Engines

- **NameNode detects DataNode failures**
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

# Data Correctness

- **Use Checksums to validate data**
  - Use CRC32
- **File Creation**
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- **File access**
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas

# Data Pipelines

- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the Client moves on to write the next block in file

# Hadoop Subprojects

- **Pig** (Initiated by Yahoo!)
  - High-level language for data analysis
- **HBase** (initiated by Powerset)
  - Table storage for semi-structured data
- **Zookeeper** (Initiated by Yahoo!)
  - Coordinating distributed applications
- **Hive** (initiated by Facebook, coming soon)
  - SQL-like Query language and Metastore
- **Mahout**
  - Machine learning

# Rebalancer

## **Goal: % disk full on DataNodes should be similar**

- Usually run when new DataNodes are added
- Cluster is online when Rebalancer is active
- Rebalancer is throttled to avoid network congestion
- Command line tool

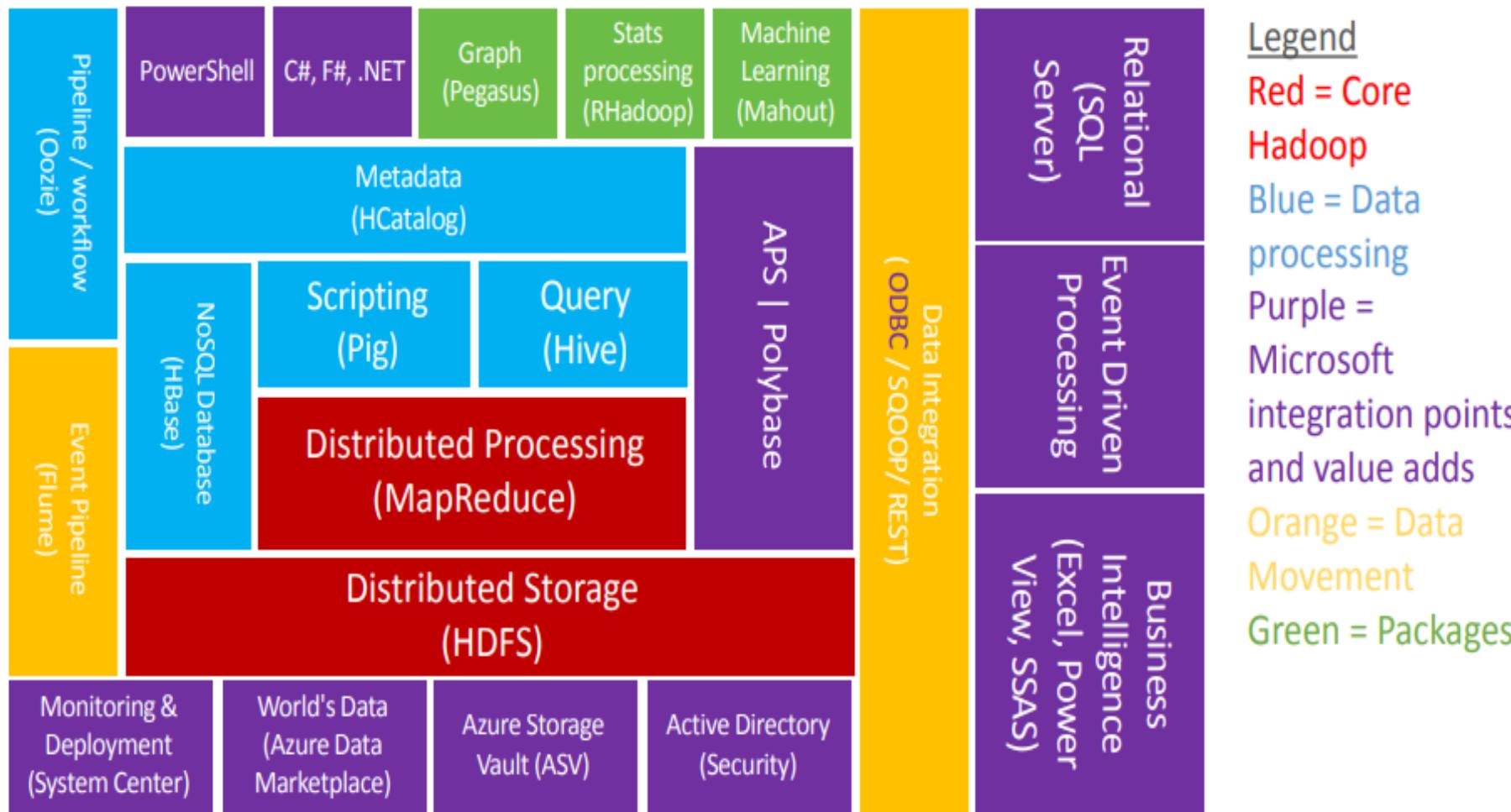
# User Interface

- Command for HDFS User:
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
- Command for HDFS Administrator
  - hadoop fsadmin -report
  - hadoop fsadmin -decommission datanodename
- Web Interface
  - <http://host:port/dfshealth.jsp>

# Parallelisms

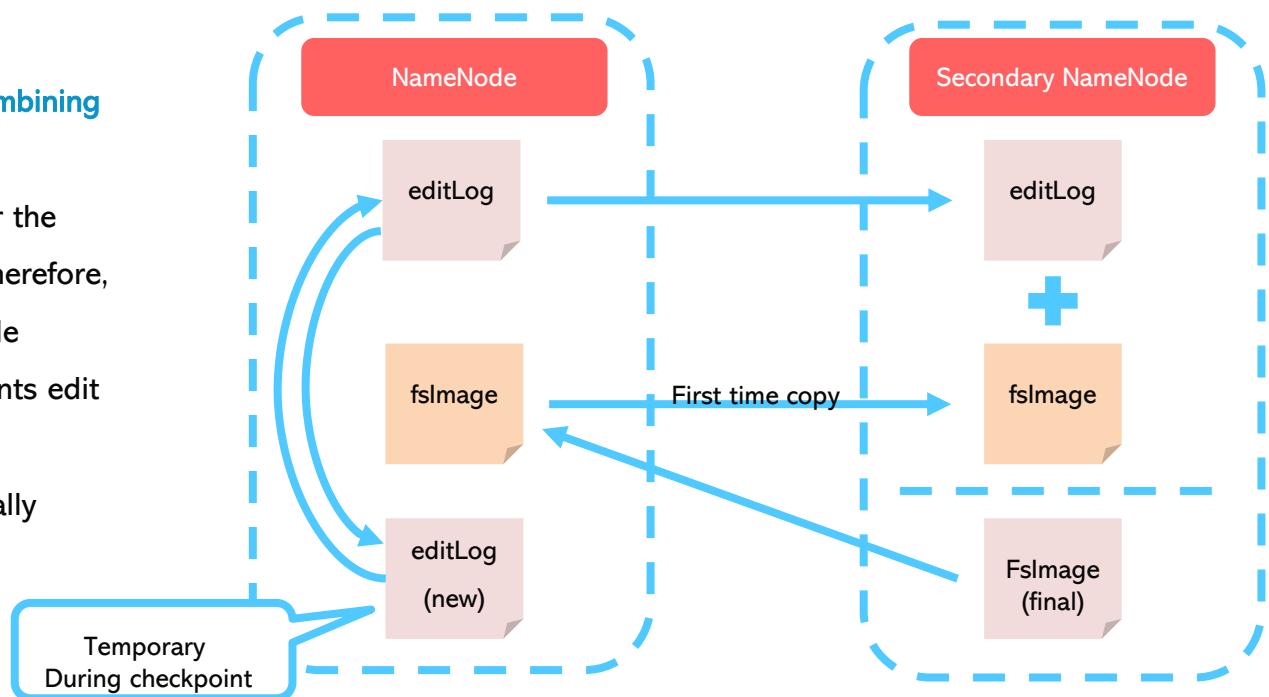
- Map is inherently parallel
  - Each list element processed independently
- Reduce is inherently sequential
  - Unless processing multiple lists
  - Grouping to produce multiple lists

# Hadoop Ecosystem



# Secondary NameNode & Checkpointing

- Checkpointing is a **process of combining edit logs with Fslimage**
- Secondary NameNode takes over the **responsibility of checkpointing**, therefore, making NameNode more available
- Allows faster Failover as it prevents edit logs from getting too huge
- Checkpointing happens periodically (default: 1 hour)



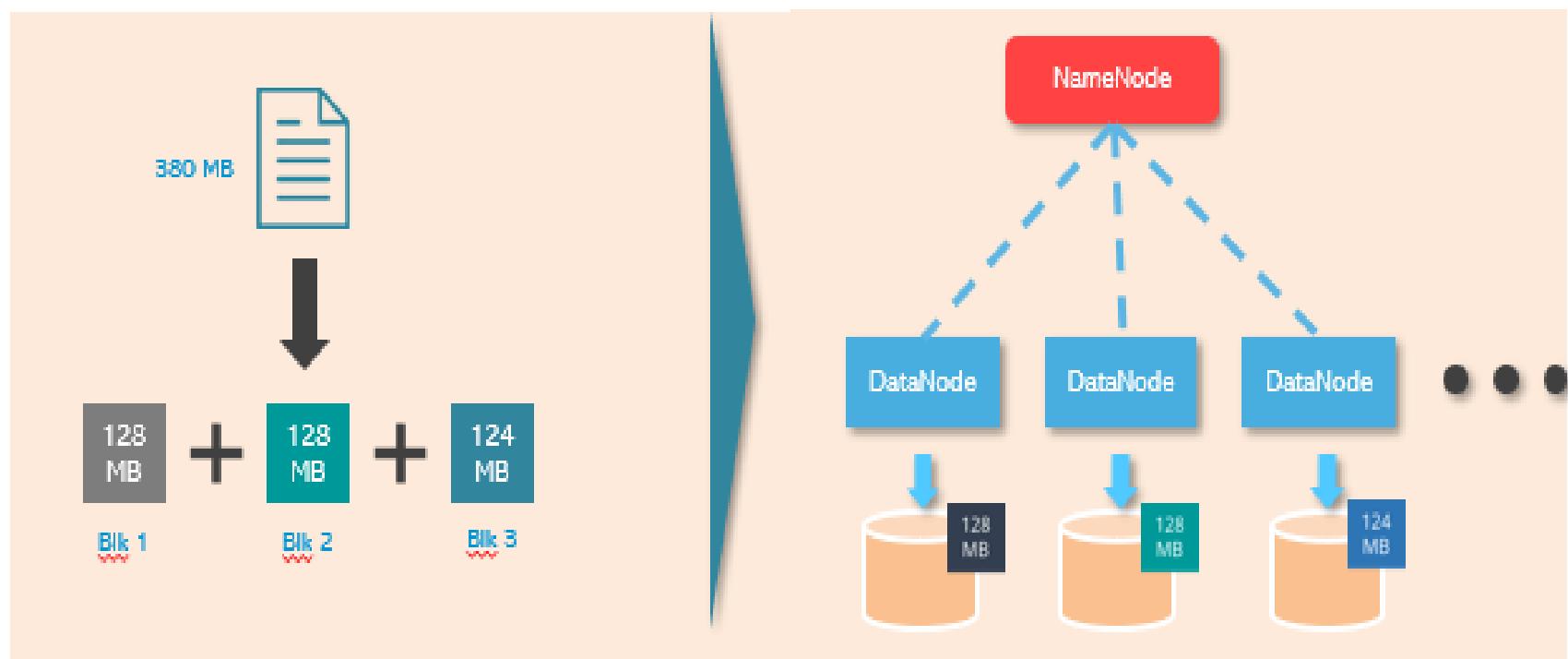
# **How The Data Is Actually Stored In Datanodes? HDFS Data Blocks**



## HDFS Data Blocks

# HDFS Data Blocks

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)

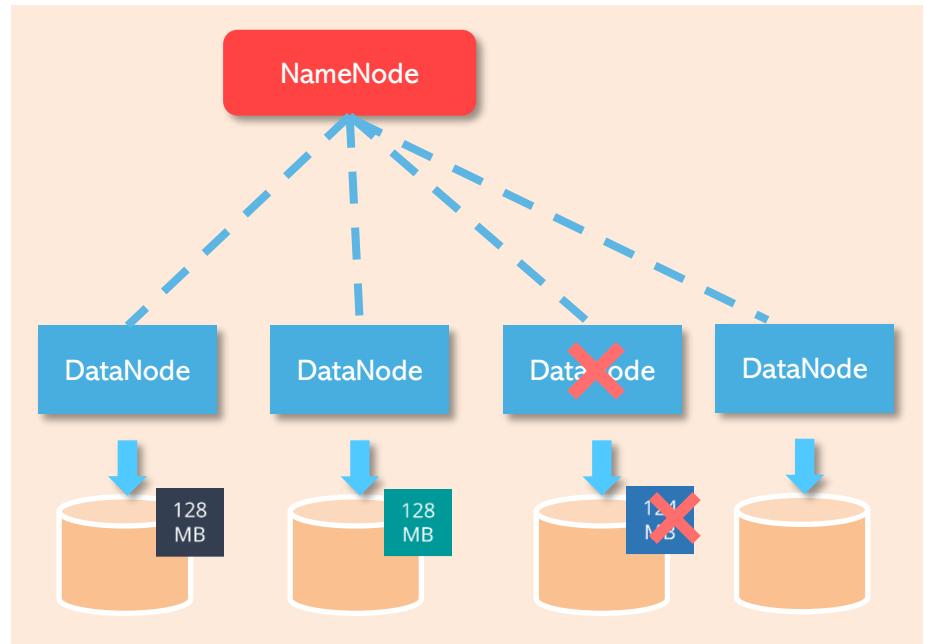
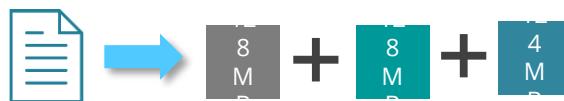


# Fault Tolerance: How Hadoop copes up with DataNode Failure?

# Fault Tolerance

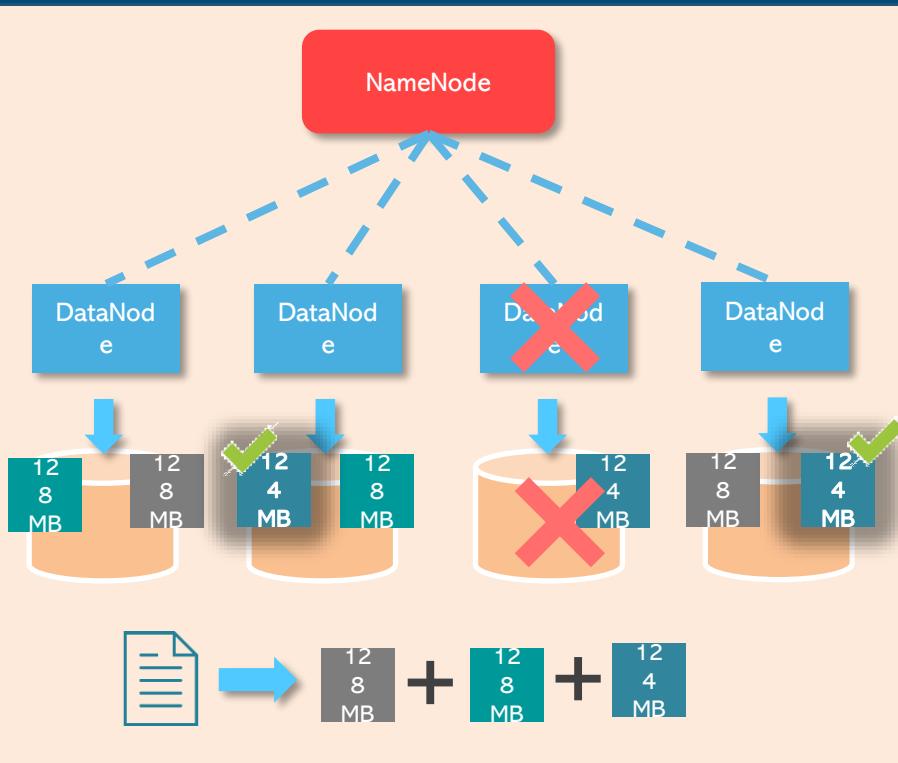
## Scenario:

One of the DataNodes crashed containing the data blocks



# Solution: Replication Factor

# Replication Factor



## Solution:

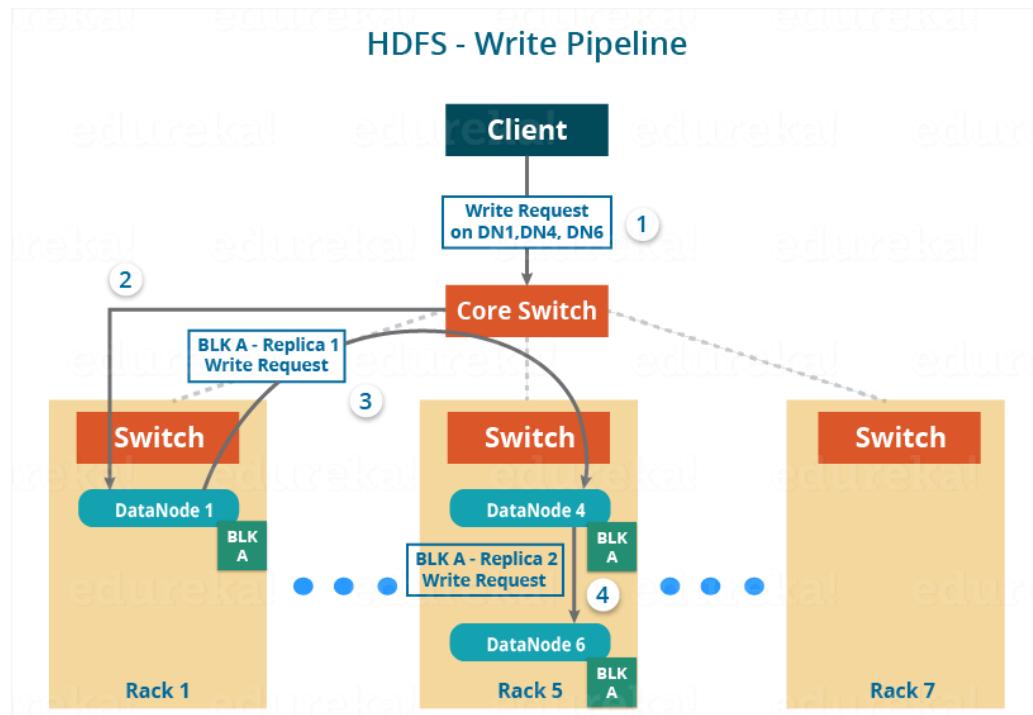
Each data blocks are replicated (thrice by default) and are distributed across different DataNodes



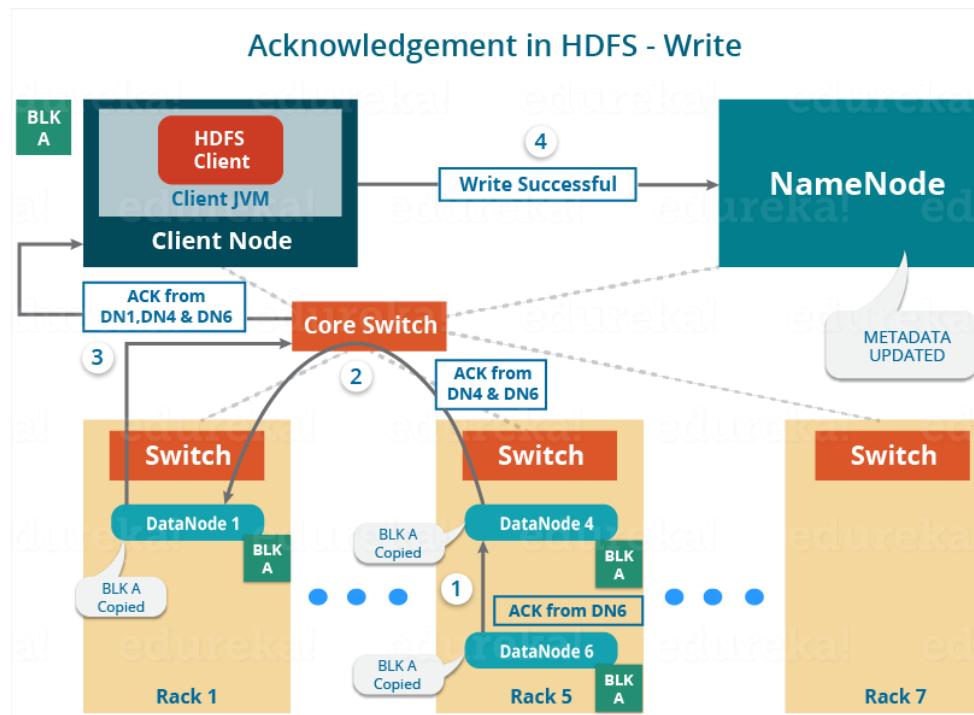
# HDFS Read & Write Mechanism

# HDFS Write Mechanism

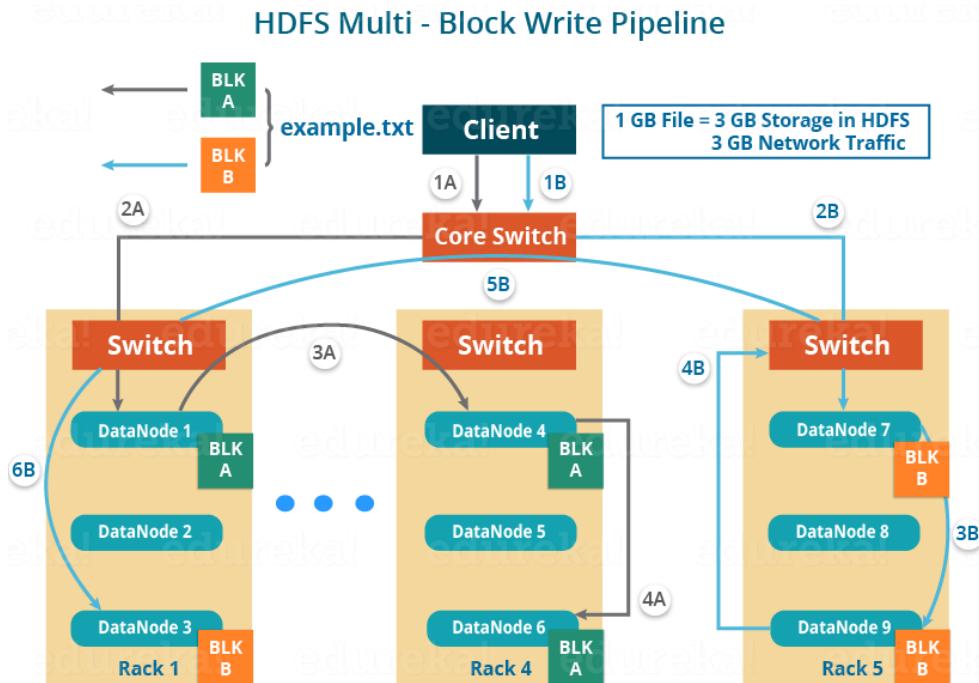
# HDFS Write Mechanism – Writing a Block



# HDFS Write Mechanism – Acknowledgement



# HDFS Multi Block Write Mechanism



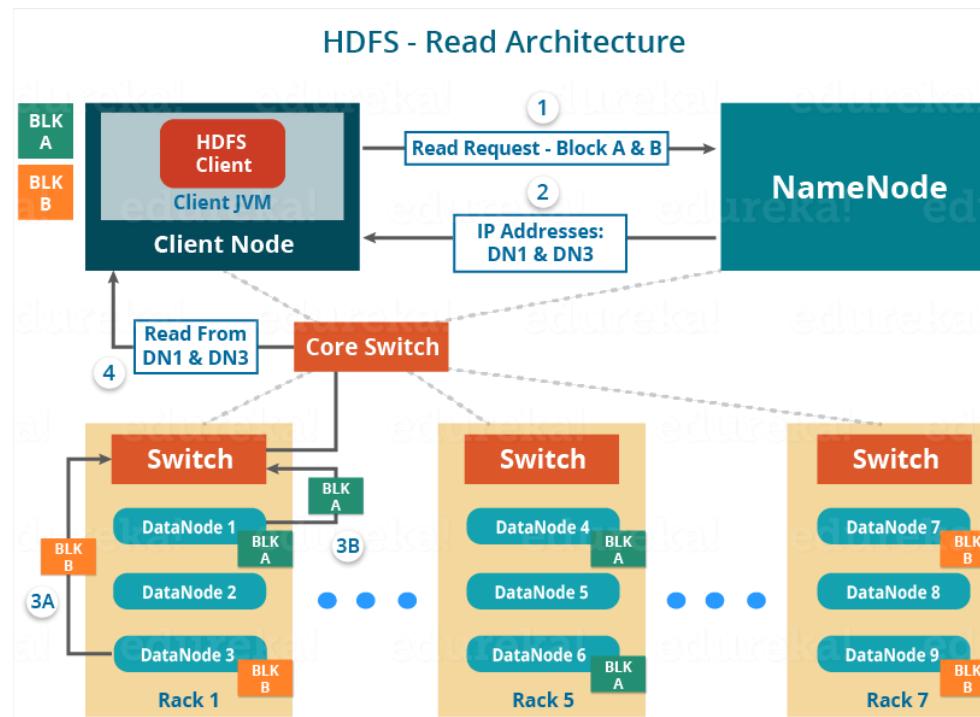
For Block A: 1A -> 2A -> 3A -> 4A

For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

# HDFS Read Mechanism

Looking for Big Data Program? Call us at **IN: 8047474382/ US: +1 (415) 980-4399** or visit [www.edureka.co](http://www.edureka.co)

# HDFS Read Mechanism





# Cloud Computing

## SEWP ZG527

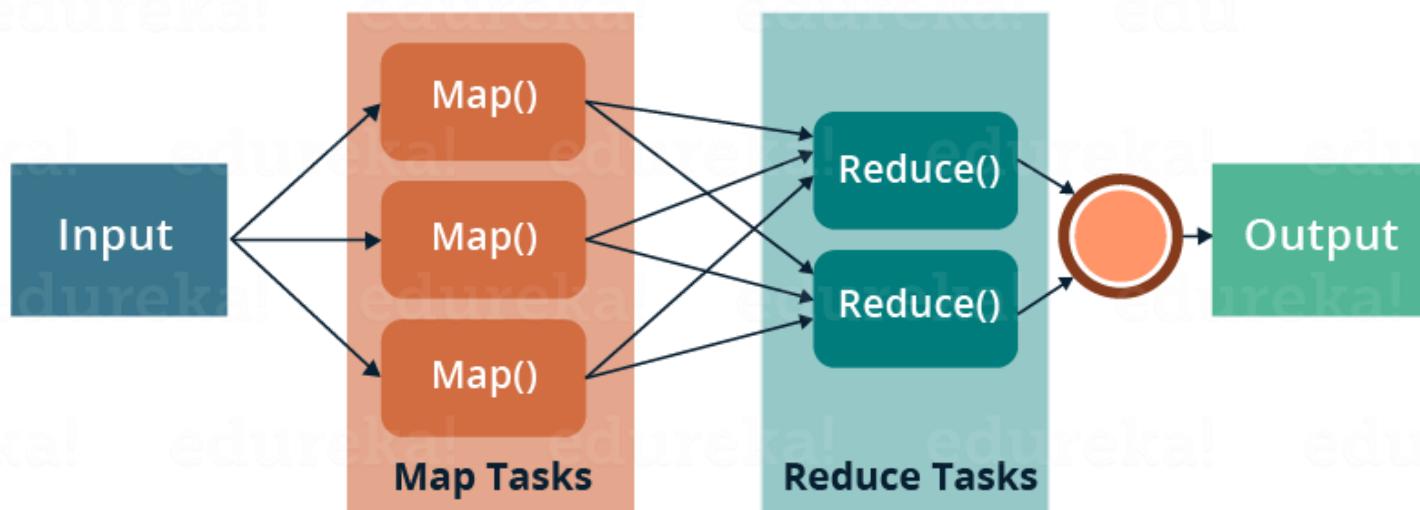
**BITS** Pilani



# What is MapReduce?

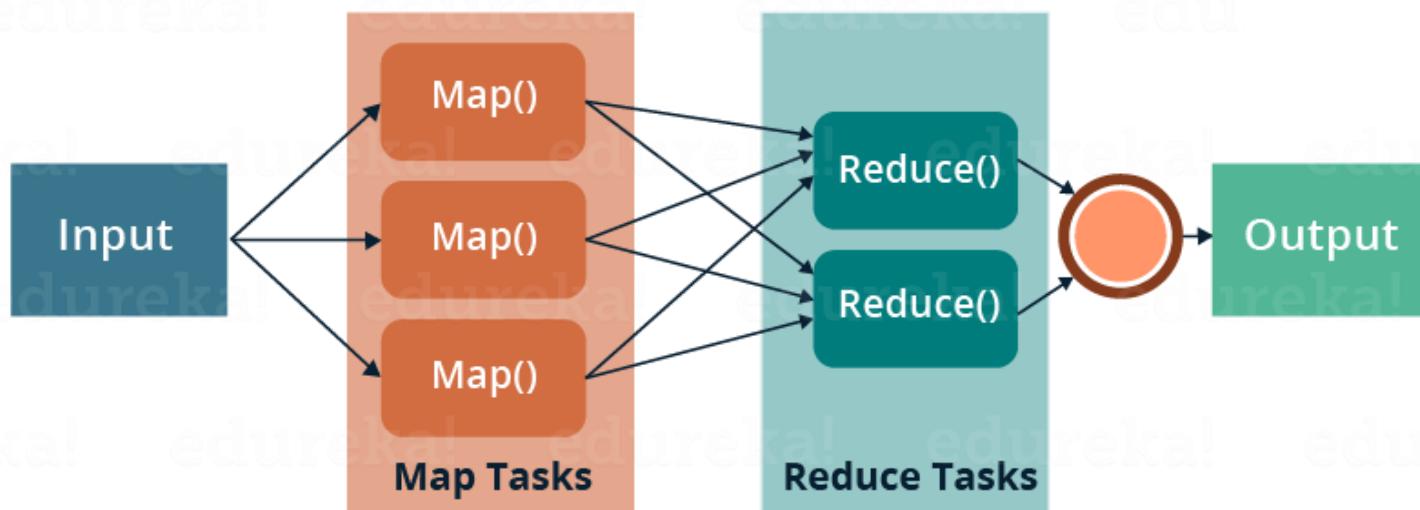
# What is MapReduce?

MapReduce is a **programming framework** that allows us to perform **distributed** and **parallel** processing on large data sets in a distributed environment



# What is MapReduce?

MapReduce is a **programming framework** that allows us to perform **distributed** and **parallel** processing on large data sets in a distributed environment

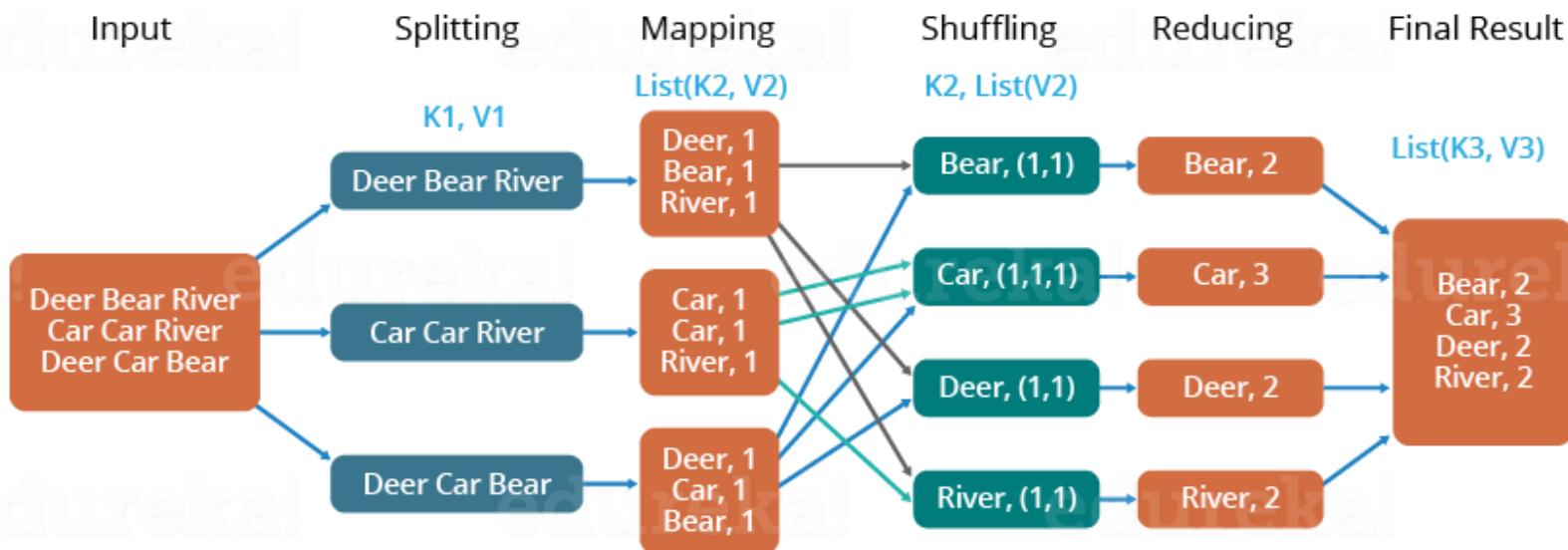




# MapReduce Word Count Program

# MapReduce Word Count Program

The Overall MapReduce Word Count Process



# MapReduce Word Count Program

Three Major Parts of MapReduce Program:

1

## Mapper Code:

You write the mapper logic over here i.e. how map task will process the data to produce the key-value pair to be aggregated

2

## Reducer Code:

You write reducer logic here which combines the intermediate key-value pair generated by Mapper to give the final aggregated output

3

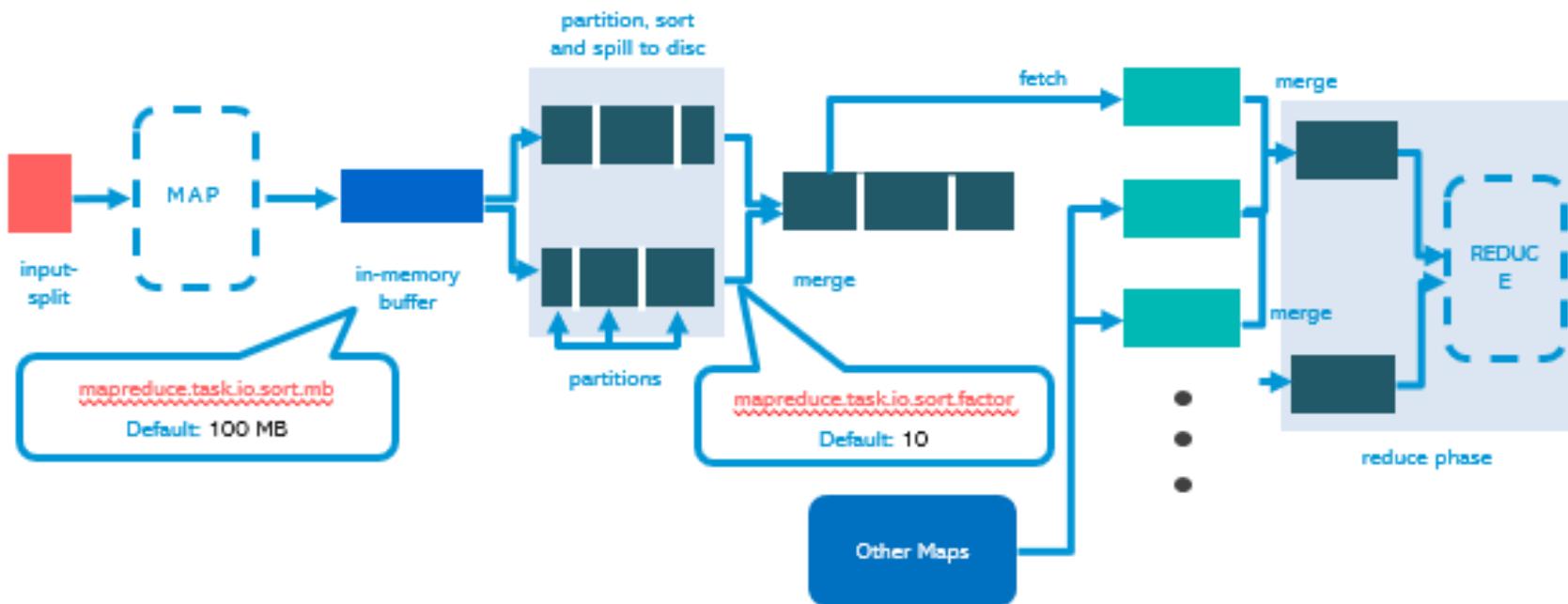
## Driver Code

You specify all the job configurations over here like job name, Input path, output path, etc.

A large, bold, blue number "10" is centered within a white circle. The circle has a thick, dark blue outline. The entire graphic is set against a solid blue background.

# MapReduce Job Workflow

# MapReduce Job Workflow



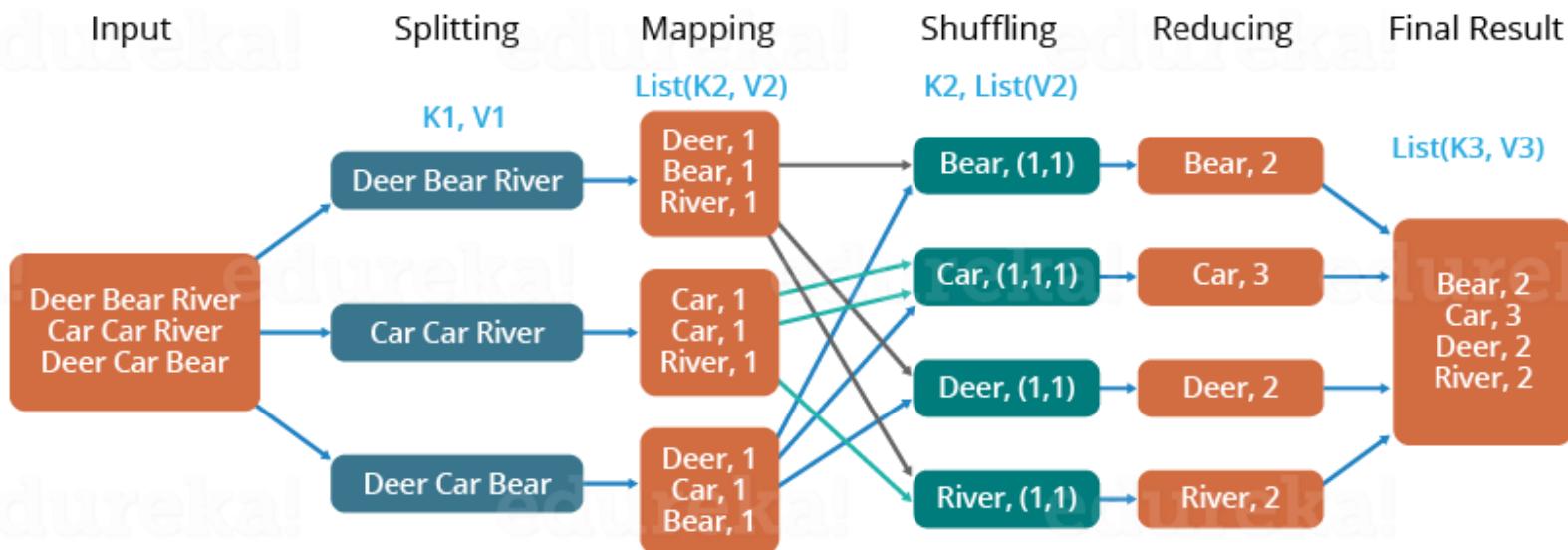


11

# MapReduce Word Count Program

# MapReduce Word Count Program

The Overall MapReduce Word Count Process



# MapReduce Word Count Program

Three Major Parts of MapReduce Program:

1

## Mapper Code:

You write the mapper logic over here i.e. how map task will process the data to produce the key-value pair to be aggregated

2

## Reducer Code:

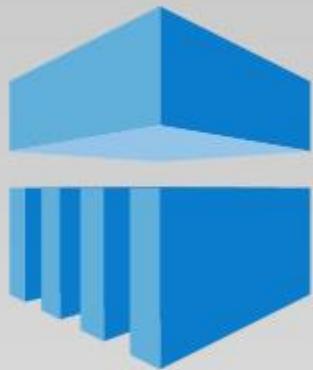
You write reducer logic here which combines the intermediate key-value pair generated by Mapper to give the final aggregated output

3

## Driver Code

You specify all the job configurations over here like job name, Input path, output path, etc.

# AWS Elastic Map Reduce



Provides a managed Hadoop framework  
Quickly & cost-effectively process vast amounts of data  
Makes it easy, fast & cost-effective for you to process data  
Run other popular distributed frameworks such as Spark

# Low Cost

Easy to Use

Elastic



# Amazon EMR

Flexible

Reliable

Secure

# AWS EMR Use Cases

## Clickstream Analysis

Amazon EMR can be used to analyze click stream data in order to segment users and understand user preferences. Advertisers can also analyze click streams and advertising impression logs to deliver more effective ads.

## Genomics

Amazon EMR can be used to process vast amounts of genomic data and other large scientific data sets quickly and efficiently. Researchers can access genomic data hosted for free on AWS.

## Log Processing

Amazon EMR can be used to process logs generated by web and mobile applications. Amazon EMR helps customers turn petabytes of un-structured or semi-structured data into useful insights about their applications or users.

# Core Characteristics of EMR



Low Cost

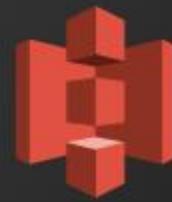
Low Hourly Pricing

Amazon EC2 Spot Integration



Amazon EC2 Reserved Instance Integration

Elasticity



Amazon S3 Integration

# Core Characteristics of EMR

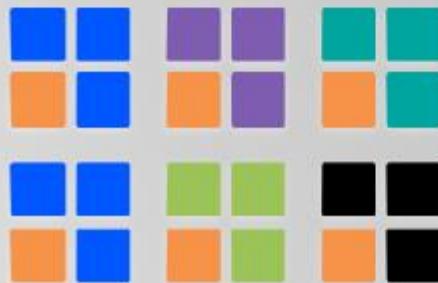


Elastic

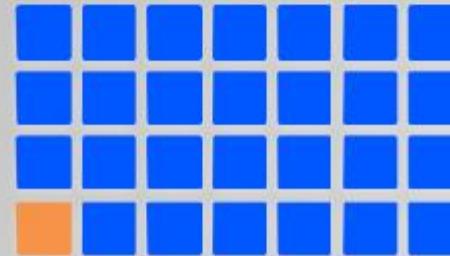
Provision as much capacity as you need

Add or remove capacity at any time

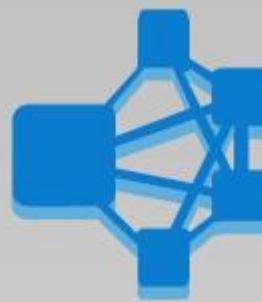
Deploy Multiple Clusters



Resize a Running Cluster



# EMR Data Stores



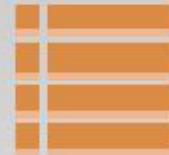
Amazon  
EMR



Amazon  
S3



Hadoop Distributed  
File System



Amazon  
DynamoDB



Amazon  
Redshift



Amazon  
Glacier



Amazon  
Relational  
Database Service

# EMR and S3

## Amazon S3 + Amazon EMR



- Allows you to decouple storage and computing resources
- Use Amazon S3 features such as server-side encryption
- When you launch your cluster, EMR streams data from S3
- Multiple clusters can process the same data concurrently

### **What is Amazon S3? Elaborate.**

S3 (Simple Storage Service) provides scalable object storage space to firms and IT professionals. It is one of the earliest services introduced by AWS. The easy-to-use web services interface of S3 allows users to store and retrieve data from remote locations. S3 contains buckets to store files/data.

Users create a bucket in the S3 and name it as if it is a universal namespace. An HTTP 200 code is received on successful uploading of a file to the assigned S3 bucket. A unique name is given to each bucket to generate the DNS address (unique).

**There are four types of pricing models for Amazon EC2 instances that are as follows:**

- **On-demand instance** – On-demand pricing or pay-as-you-go model allows you to pay only for the resources used till now. Depending on the instances, you will have to pay by second/hour for the resources. The on-demand pricing model is good if the work hours are short and unpredictable as they do not require any upfront payment.
- **Reserved instance** – It is the best model to use if you have a prerequisite for your upcoming requirements. Firms calculate their future EC2 requirements and pay upfront to get a discount of up to 75%. Reserved instances will save computing capacity for you, and you can use them wherever required.
- **Spot Instance** – If some extra amount of computing capacity is required immediately, one can opt for spot instances at up to a 90% discount. The unused computing capacity is sold at a heavily discounted rate via the spot instance pricing model.
- **Dedicated hosts** – A customer can reserve a physical EC2 server by opting for the dedicated hosts pricing model.

## **What are the characteristics of cloud architecture that separates it from the traditional one?**

The characteristics that make **cloud architecture** above traditional architecture is

- - According to the demand, cloud architecture provides the hardware requirement
  - Cloud architecture is capable of scaling the resource on demand
  - Cloud architecture is capable of managing and handling dynamic workloads without failure

## **What are the different types of virtualization in AWS, and what are the differences between them?**

The three major types of virtualization in AWS are:

- ***Hardware Virtual Machine (HVM)***

It is a fully virtualized hardware, where all the virtual machines act separate from each other. These virtual machines boot by executing a master boot record in the root block device of your image.

- ***Paravirtualization (PV)***

Paravirtualization-GRUB is the bootloader that boots the PV AMIs. The PV-GRUB chain loads the kernel specified in the menu.

- ***Paravirtualization on HVM***

PV on HVM helps operating systems take advantage of storage and network I/O available through the host.

## **Question Cloud interoperability and its use case**

### **What are the main drawbacks of Docker?**

Some notable drawbacks of Docker are:

- Doesn't provide a storage option
- Offer a poor monitoring option.
- No automatic rescheduling of inactive Nodes
- Complicated automatic horizontal scaling set up

### **What are the common instruction in Dockerfile?**

The common instruction in Dockerfile are: FROM, LABEL, RUN, and CMD

## **What are Docker Namespaces?**

The Namespace in Docker is a technique which offers isolated workspaces called the Container. Namespaces also offer a layer of isolation for the Docker containers.

### **Explain in detail single and multi tenant architecture**

### **What are different layers in iaas, paas and SaaS model**

## **State the limitations of virtualization.**

1. If the CPU does not allow for hardware virtualization we can run some operating system in software virtualization but it is generally slower. Some operating system will not run in software virtualization and require to have CPU with hardware virtualization so it would cost more if CPU with hardware virtualization is not possible.
2. Some of the limitations are in analysis and planning which problems can be divided into three types they are
  - a. It has a high risk in physical fault.
  - b. It is more complicated to set up and manage virtual environment with high critical servers in a production environment. It is not easy as managing physical servers.
3. It does not support all applications.

## **discuss the design considerations for storage network.**

The best storage area network design for a customer will take into consideration a number of critical issues:

- ❑ Uptime and availability
- ❑ Capacity and scalability
- ❑ Security
- ❑ Replication and disaster recovery

**Give the best example of open source Cloud Computing.**

**Open-source cloud** is a cloud service or solution built using **open-source software** and technologies. This includes any public, private or hybrid **cloud** model providing SaaS, IaaS, PaaS, or XaaS built and operated entirely on **open-source technologies**.

The best example of open source Cloud Computing is **OpenStack and Nebula**.

This is one of the most frequently asked cloud computing interview questions.

Cloud computing lets us store and access our applications or data over remote computers instead of our computer. First of all, the cloud is just a metaphor for technology. Cloud data centers might be anywhere globally; we can also access them from anywhere with an Internet-connected device. It has the following benefits as given below:

**Pay-per-use model:** We only have to pay for the services we use.

**24/7 Availability:** It is always online! There is no such time when you simply cannot use our cloud service; you'll use it whenever you want.

**Easily Scalable:** it's effortless to proportion and down or turn off as per customers' needs. For instance, if your website's traffic increases only on Friday nights, you can opt for scaling up your servers that particular day and then scaling down for the rest of the week.

**Security:** Cloud computing offers excellent data security. Especially if the data is mission-critical, then that data can be wiped off from local drives and kept on the cloud only for your access to stop it from ending up in the wrong hands.

**Easily Manageable:** You only have to pay subscription fees; the Cloud Provider entirely maintains all maintenance, up-gradation, and delivery of services. This is backed by the Service-level Agreement (SLA).

## **Name some of the AWS services that are not region-specific**

AWS services that are not region-specific are:

- [IAM](#)
- Route 53
- Web Application Firewall
- CloudFront.

## **Name advantages and disadvantages of using serverless components in cloud computing.**

Your applicants may mention some of the disadvantages listed below when responding to this question:

- The components are not always suitable for high-performance computing
- The components may be more vulnerable to security issues
- The components may make debugging a challenge

## **Explain briefly about virtual threats.**

Some threats to virtualized systems are general in nature, as they are inherent threats to all computerized systems (such as denial-of-service, or DoS, attacks). Other threats and vulnerabilities, however, are unique to virtual machines. Many VM vulnerabilities stem from the fact that vulnerability in one VM system can be exploited to attack other VM systems or the host systems, as multiple virtual machines share the same physical hardware.

Some of the vulnerabilities exposed to any malicious-minded individuals regarding security in virtual environments:

**Shared clipboard** — Shared clipboard technology allows data to be transferred between VMs and the host, providing a means of moving data between malicious programs in VMs of different security realms.

**Keystroke logging** — Some VM technologies enable the logging of keystrokes and screen updates to be passed across virtual terminals in the virtual machine, writing to host files and permitting the monitoring of encrypted terminal connections inside the VM.

**VM monitoring from the host** — because all network packets coming from or going to a VM pass through the host, the host may be able to affect the VM by the following:

1. Starting, stopping, pausing, and restart VMs.
2. Monitoring and configuring resources available to the VMs, including CPU, memory, disk, and network usage of VMs.
3. Adjusting the number of CPUs, amount of memory, amount and number of virtual disks and number of virtual network interfaces available to a VM.
4. Monitoring the applications running inside the VM.
5. Viewing, copying, and modifying data stored on the VM's virtual disks.

**Virtual machine monitoring from another VM** — Usually, VMs should not be able to directly access one another's virtual disks on the host.

**Virtual machine backdoors** — a backdoor, covert communications channel between the guest and host could allow intruders to perform potentially dangerous operations.

**Mention some optimization strategies in the cloud.**

- Identify resources that are not in use.
- Merge all idle resources.
- Right-size your computing resources.
- Choose the appropriate storage types.
- Invest in reserved instances for long term cloud resources usage.
- Leverage spot instances for short term cloud usage.
- Delete unused elastic block store(EBS) snapshots

## **Can NameNode and DataNode be a commodity hardware?**

The smart answer to this question would be, DataNodes are commodity hardware like personal computers and laptops as it stores data and are required in a large number. But from your experience, you can tell that, NameNode is the master node and it stores metadata about all the blocks stored in HDFS. It requires high memory (RAM) space, so NameNode needs to be a high-end machine with good memory space.

## **Why do we use HDFS for applications having large data sets and not when there are a lot of small files?**

HDFS is more suitable for large amounts of data sets in a single file as compared to small amount of data spread across multiple files. As you know, the NameNode stores the metadata information regarding the file system in the RAM. Therefore, the amount of memory produces a limit to the number of files in my HDFS file system. In other words, too many files will lead to the generation of too much metadata. And, storing these metadata in the RAM will become a challenge. As a thumb rule, metadata for a file, block or directory takes 150 bytes.

## **The role of cloud computing in social networking has many dimensions.**

- Cloud computing vendors such as Amazon and Salesforce nowadays provide various services, including Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM). These services are delivered through cloud servers. Hence, clients can use the system's scalability and flexibility, whereas they don't need to purchase standalone hardware or software.

Big data analysis for social sites is another face of utilizing cloud computing besides data storage. Business users can get more improved analytics through this.

- From a disaster recovery perspective, the cloud is a safer data storage option. Social networks can do data backup and data recovery at a reduced cost in the cloud. Additionally, data stored at a particular location is riskier than saving it to the cloud. When your data is in the cloud, there are no hardships encountered during recovery. Also, social network users can access shared resources from anywhere using cloud computing. No doubt, this is a beneficial option for most social networks as they hold personal information of its clients

Q How is Docker different from Hypervisor?

Q. Write the name of top ten obstacles and opportunities for adoption and growth of cloud computing?

Q What is the difference between traditional data centers and the cloud?

Q . Name some of the AWS services that are not region-specific

Q What are the characteristics of Big Data?

Q Suppose Hadoop spawned 100 tasks for a job and one of the task failed. What will Hadoop do? Explain in detail

Q . **Can NameNode and DataNode be a commodity hardware?**

Q . What is cloud interoperability and Categories of Cloud Computing Interoperability and portability (marks 6)

Q . What are active and passive “NameNodes”?

Q What are the main drawbacks of Docker?

Q What is rack algorithm in hadoop

Q . Describe the cloud capacity management challenges

Q. What are the key enabling technologies in cloud computing?

Q . Define and explain the three basic types of cloud services and the AWS products that are built based on them?

Q . What are the layers of IaaS Architecture?

Q What are Problems with small files and HDFS?

Q. What are the common instruction in Dockerfile?

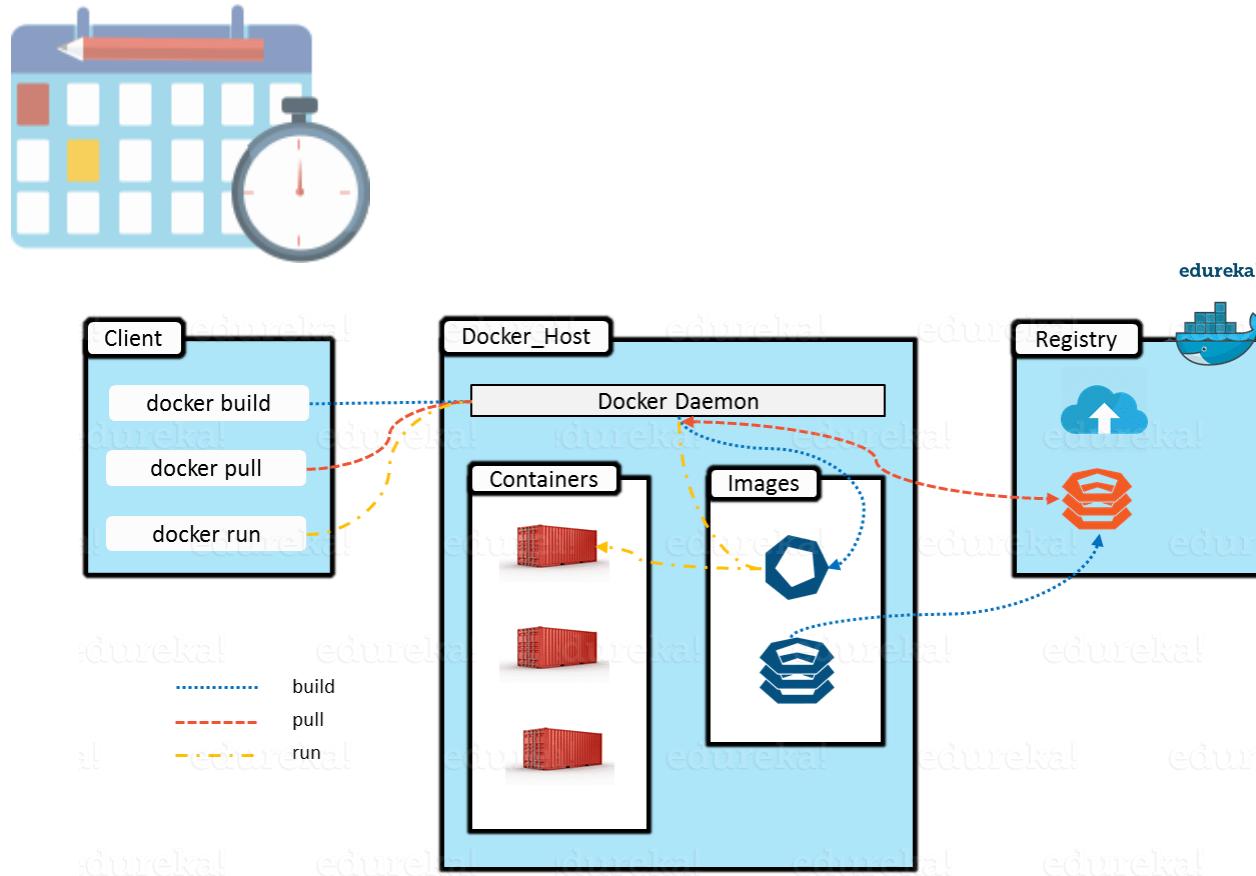
Q. Write a Docker file to create and copy a directory and built it using Tomcat server?

Q. What are Docker Namespaces?

Q . What is the need for cloud capacity management?

## What is Docker Architecture?

Docker Architecture includes a Docker client – used to trigger Docker commands, a Docker Host – running the Docker Daemon and a Docker Registry – storing Docker Images. The Docker Daemon running within Docker Host is responsible for the images and containers.

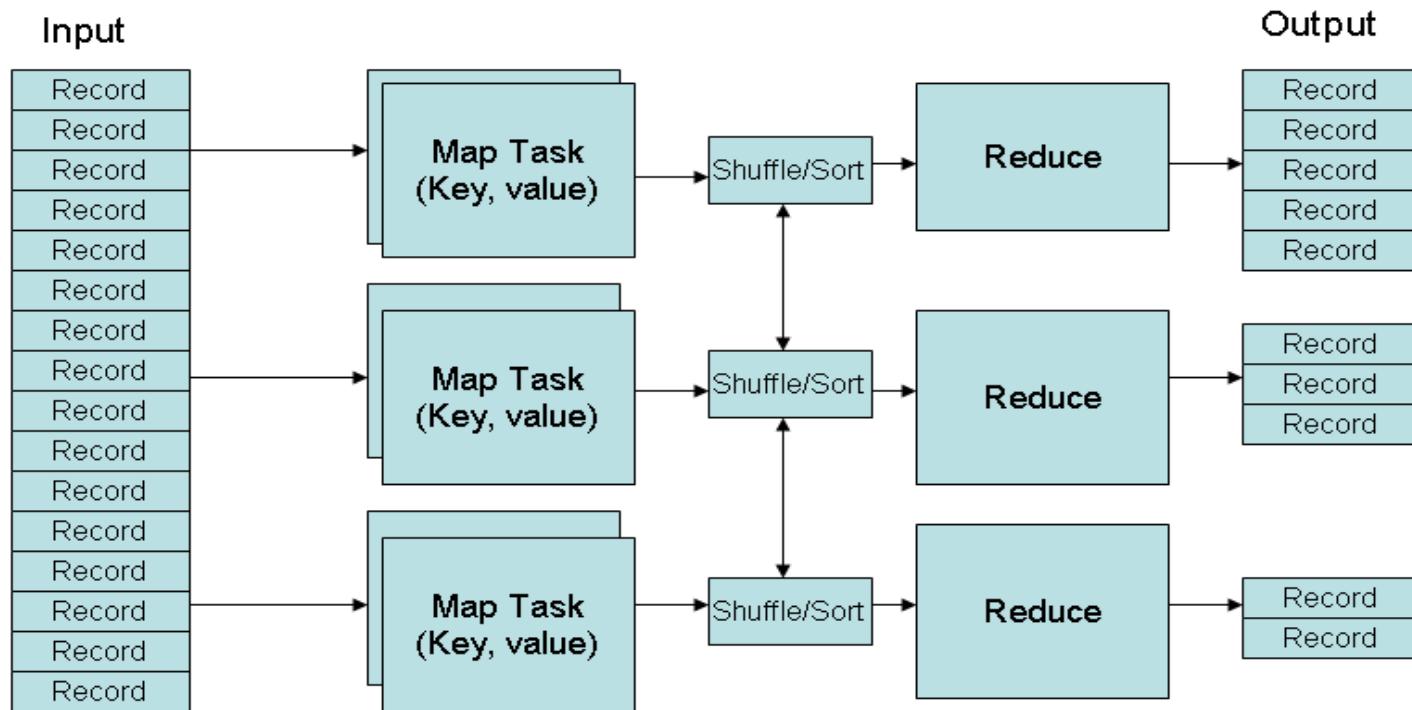


- To build a Docker Image, we can use the CLI (client) to issue a build command to the Docker Daemon (running on Docker\_Host).
- The Daemon will then build an image based on our inputs and save it in the Registry, which can be either Docker hub or a local repository
- If we do not want to create an image, then we can just pull an image from the Docker hub, which would have been built by a different user
- Finally, if we have to create a running instance of my Docker image, we can issue a run command from the CLI, which will create a Container

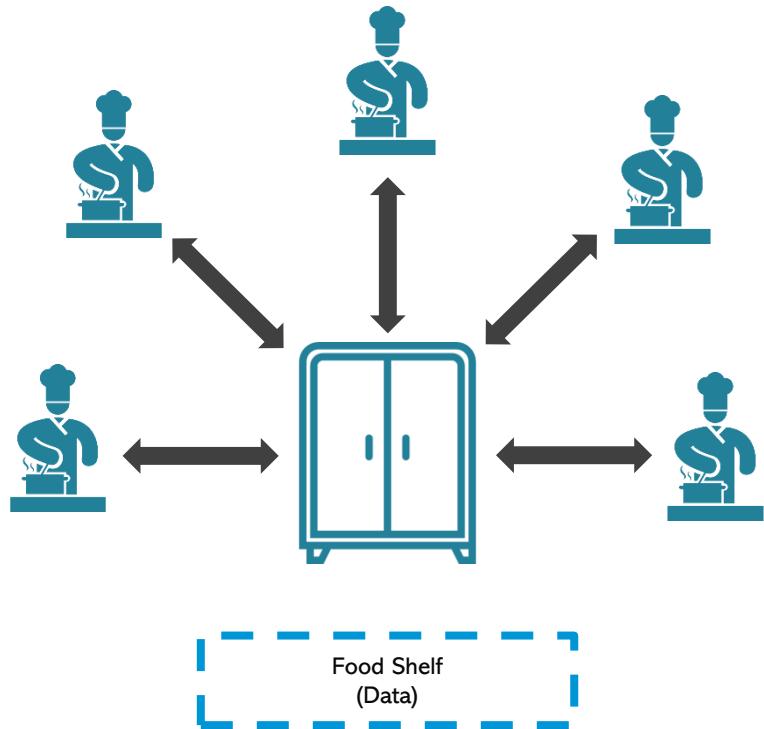
## **What is the difference between traditional data centers and the cloud?**

Traditional Data Center	Cloud Data Center
Physically present, on-premise, and tangible and accessible.	Remotely located, off-premise and intangible.
Businesses pay capital expenditures for the acquisition of hardware and software resources and operational expenditures like those for usage, maintenance, and repair of resources.	Businesses pay only the operational cost for using the resources being used.
Management and administration of the resources are the responsibility of personnel within the business.	Management and administrative responsibilities are the responsibilities of the cloud providers.
Multiple challenges to scaling, like availability of resources and latency in the acquisition process due to the procedures involved	Completely and almost instantly scalable as per requirements.
Failure may lead to loss of business as the responsibility of downtime and repair of services is entirely on the business, affecting the reliability of services.	A cloud service provider is responsible and trusted to replace the resources due to its promise of reliability and availability.

# MapReduce Phases



# Need an Effective Solution



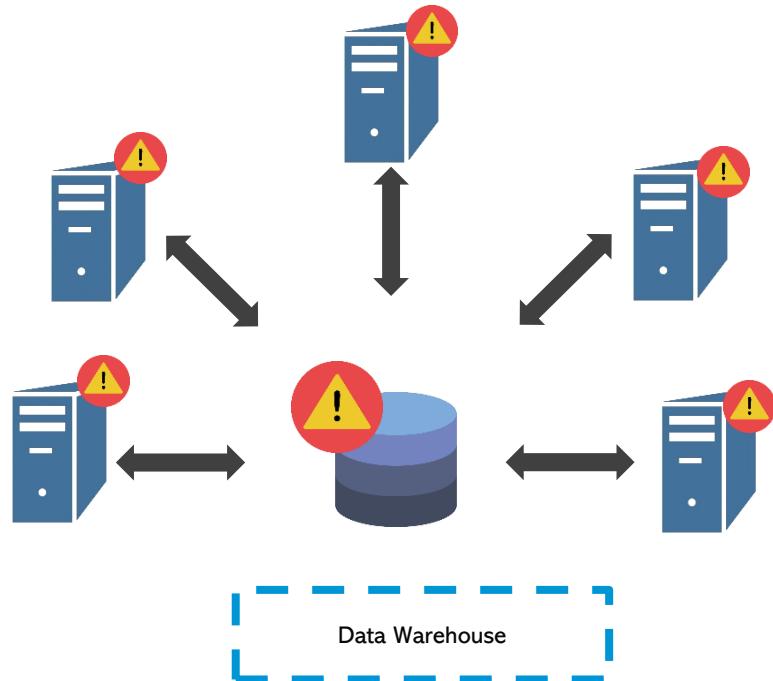
Scenario:

Multiple Cook cooking food

Issue:

Food Shelf becomes the BOTTLENECK

# Need an Effective Solution



Scenario:

Multiple Processing Unit for data processing

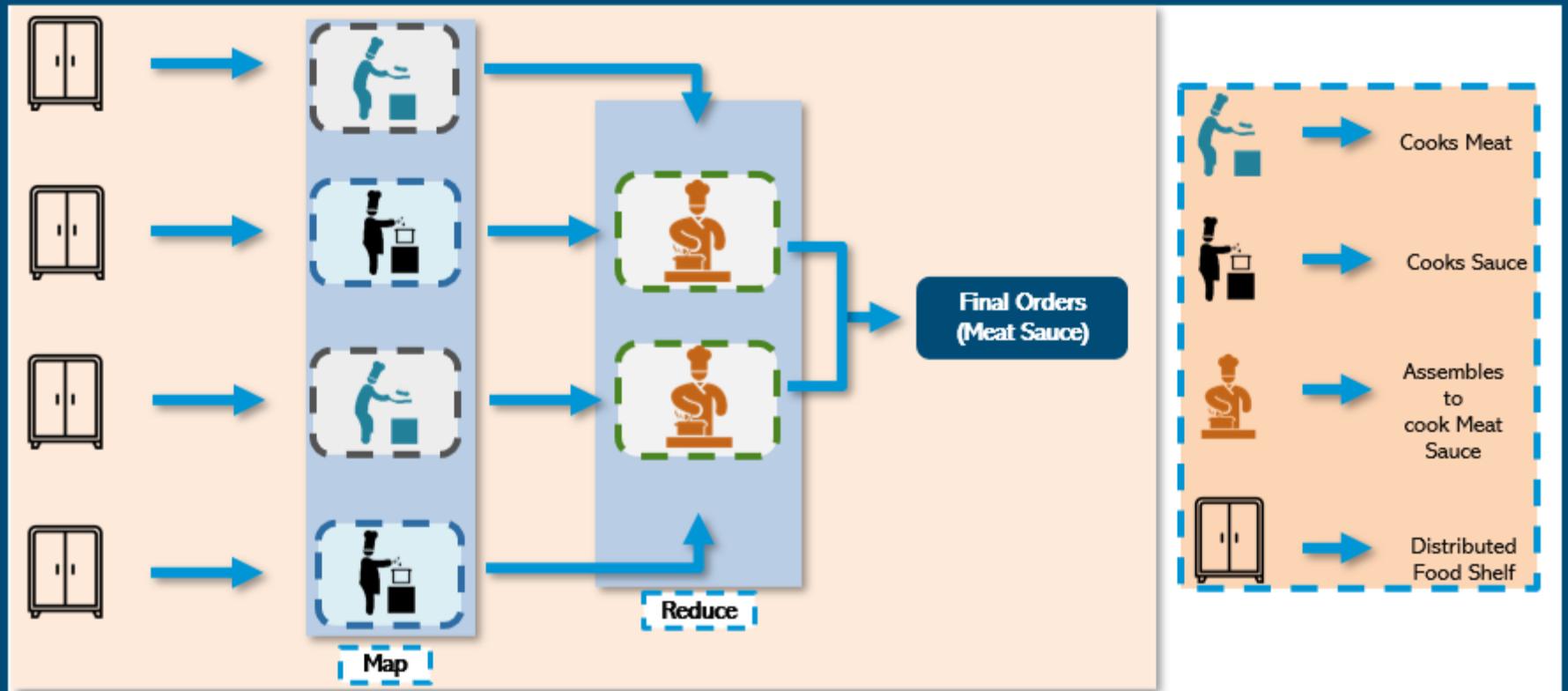
Issue:

Bringing data to processing generated lots of  
Network overhead

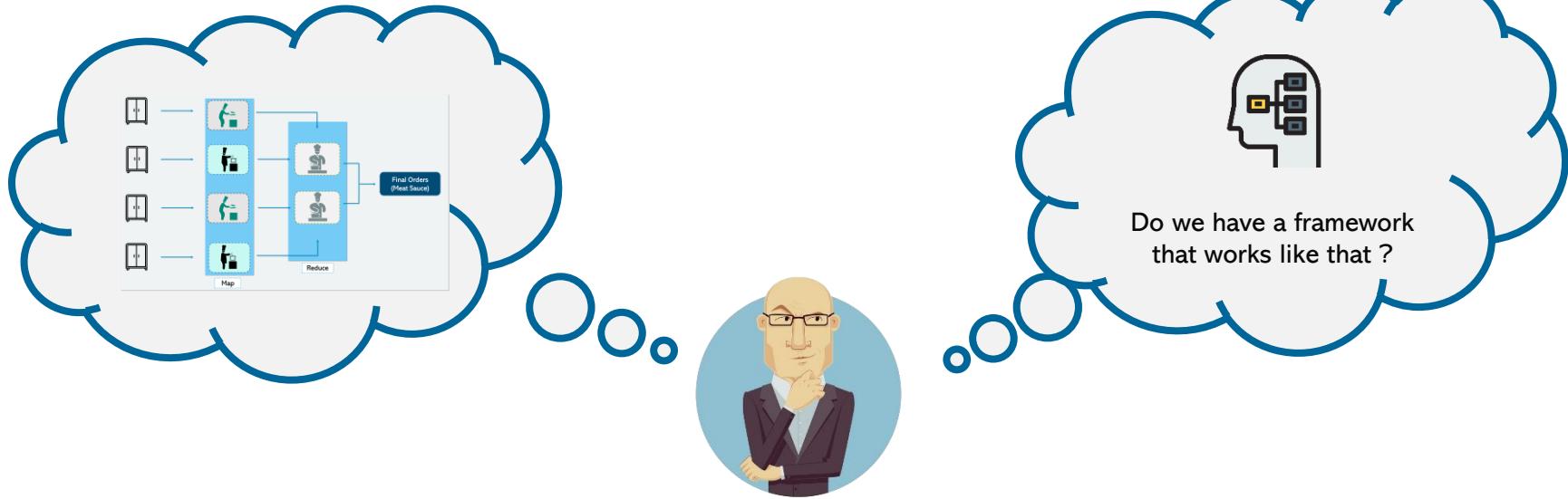
**Issue 2: Food Shelf becomes the Bottleneck**

**Solution: Distributed and Parallel Approach**

# Effective Solution



# Need a Framework

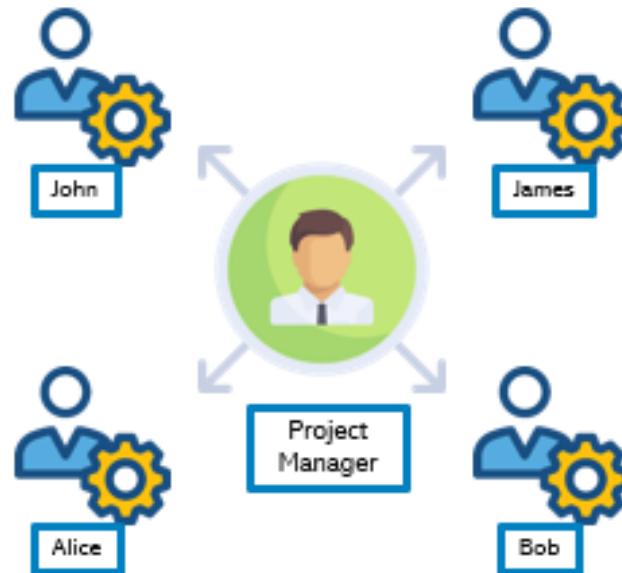


# Hadoop: Master/Slave Architecture

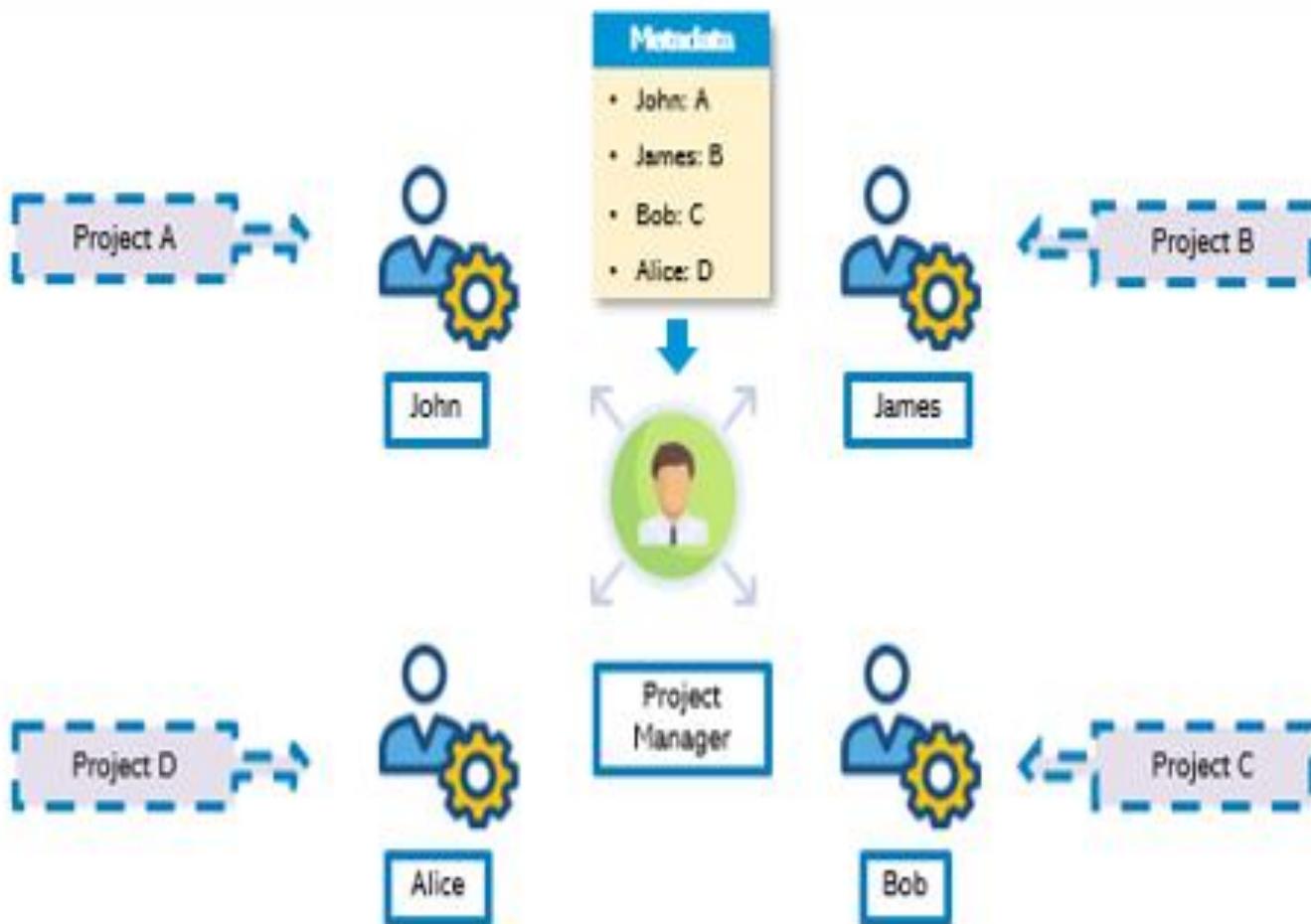
## Scenario:

A project Manager managing a team of four employees.

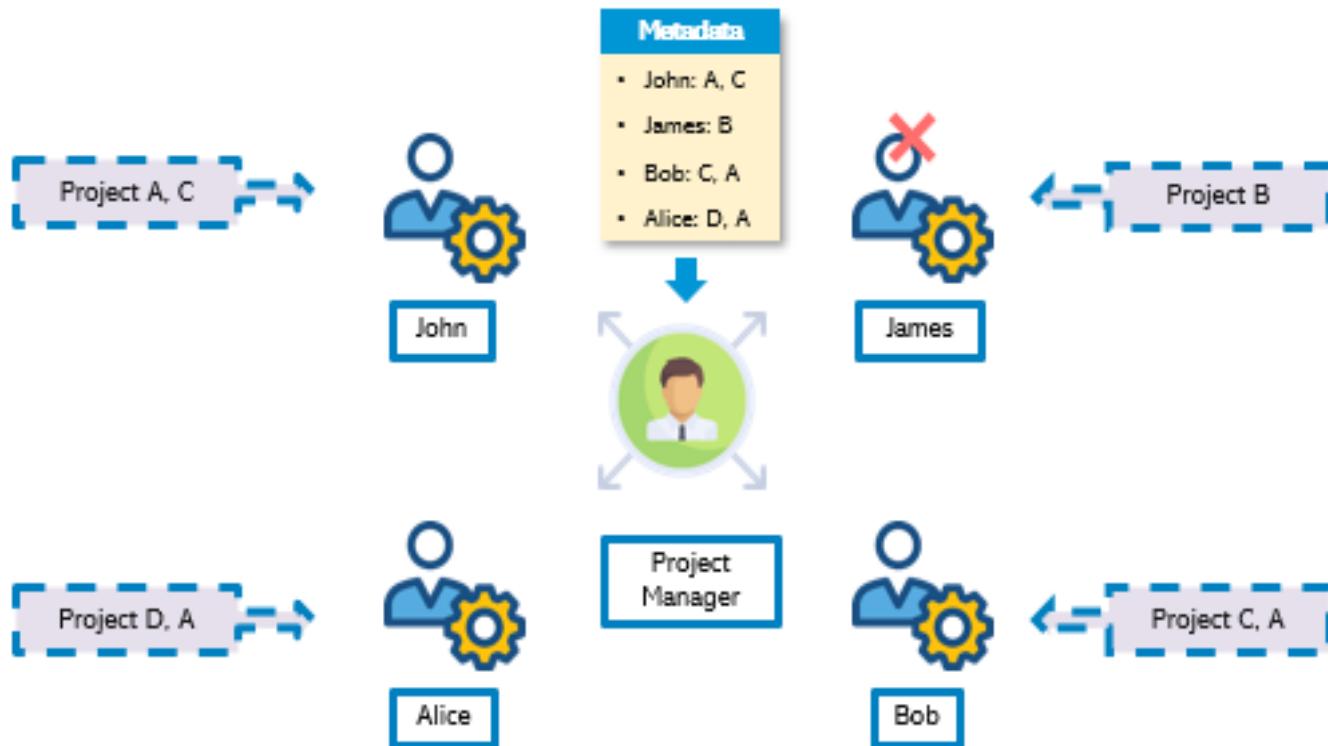
He assigns project to each of them and tracks the progress



# Hadoop: Master/Slave Architecture

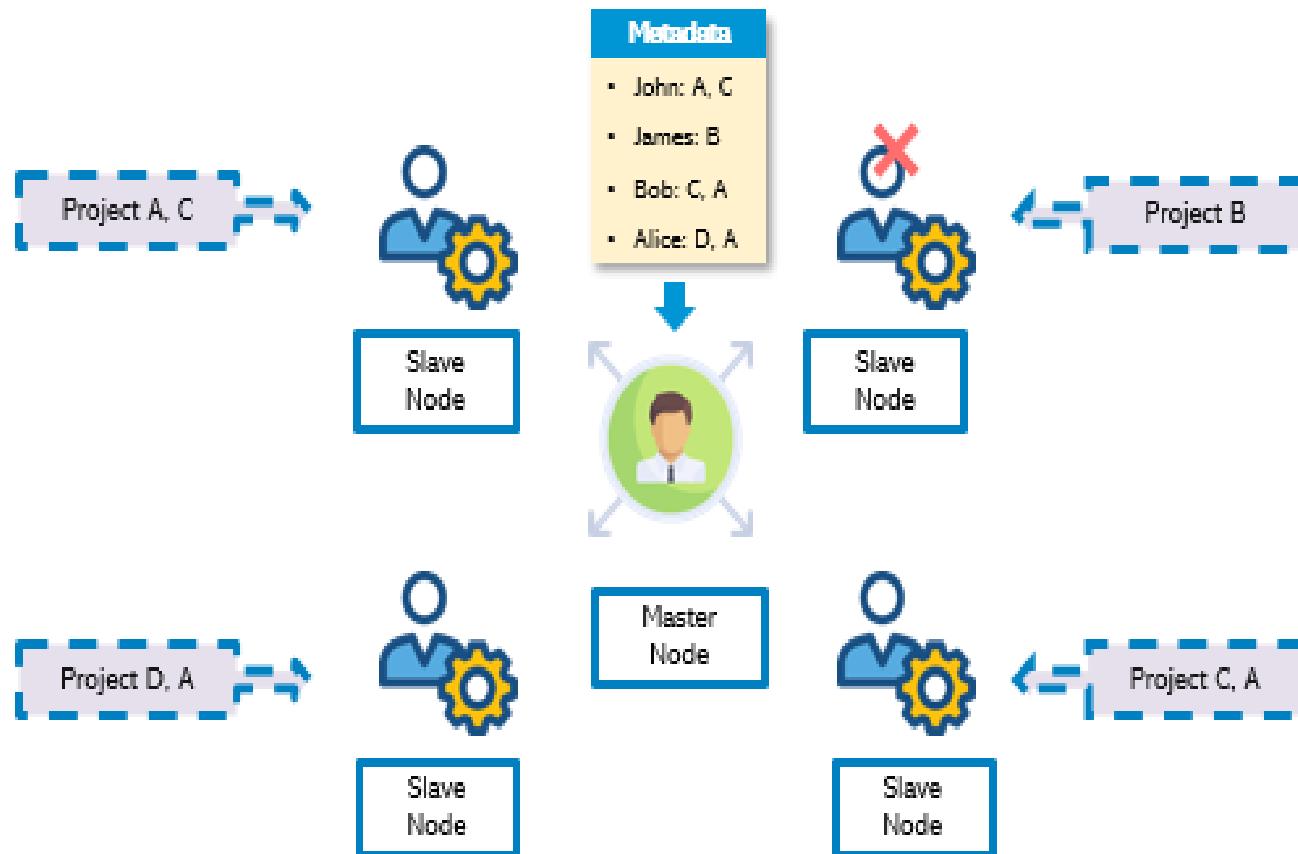


# Hadoop: Master/Slave Architecture

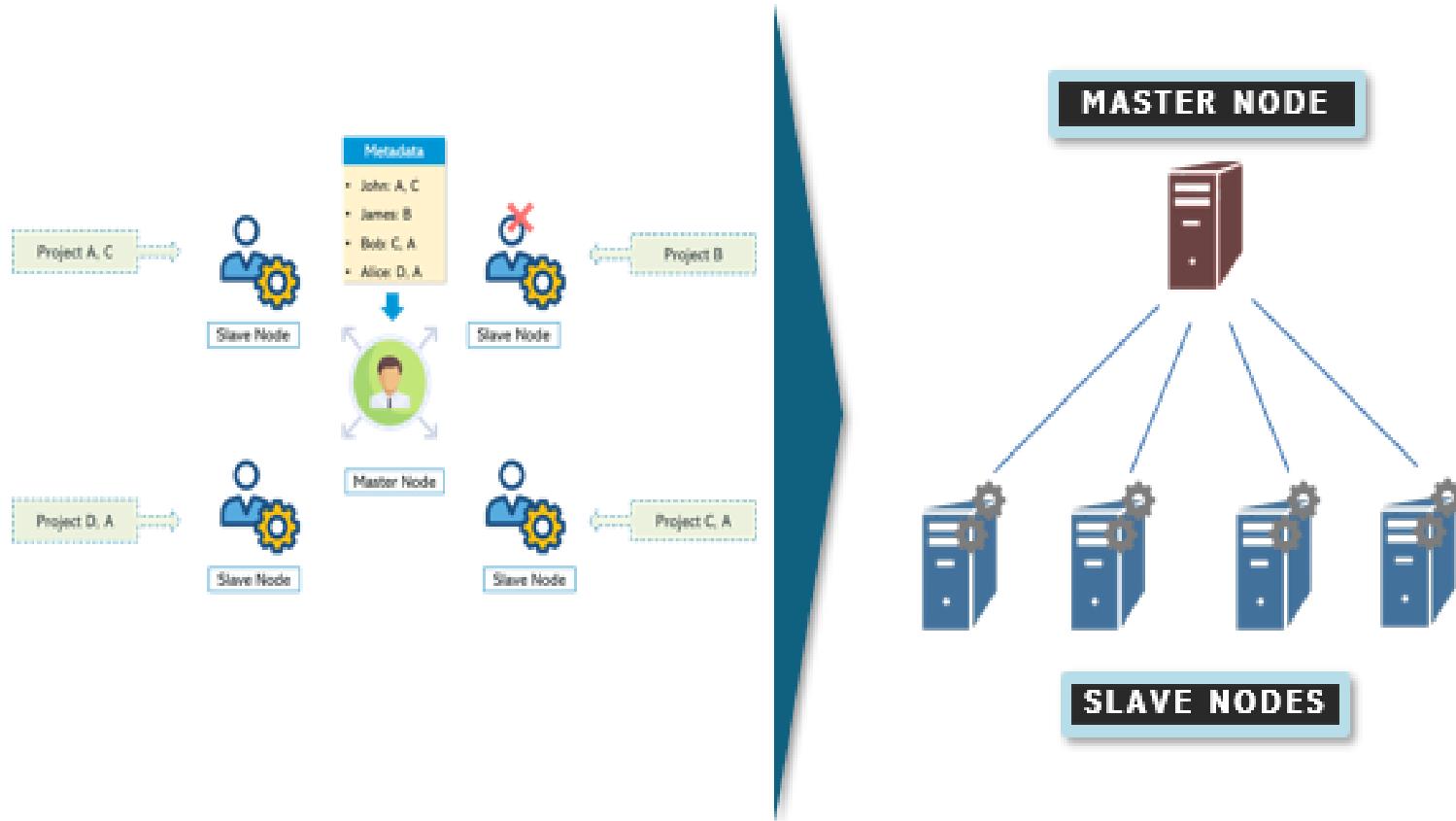


**Let us understand Problems with Big  
Data and Traditional System with a  
Story**

# Hadoop: Master/Slave Architecture



# Hadoop: Master/Slave Architecture



## Useful Links

- **HDFS Design:**
  - [http://hadoop.apache.org/core/docs/current/hdfs\\_design.html](http://hadoop.apache.org/core/docs/current/hdfs_design.html)
- **Hadoop API:**
  - <http://hadoop.apache.org/core/docs/current/api/>

# Hadoop Map/Reduce

- **The Map-Reduce programming model**
  - Framework for distributed processing of large data sets
  - Pluggable user code runs in generic framework
- **Common design pattern in data processing**

```
cat * | grep | sort | unique -c | cat > file  
input | map | shuffle | reduce | output
```
- **Natural for:**
  - Log processing
  - Web search indexing
  - Ad-hoc queries

# About Key-Value Pairs

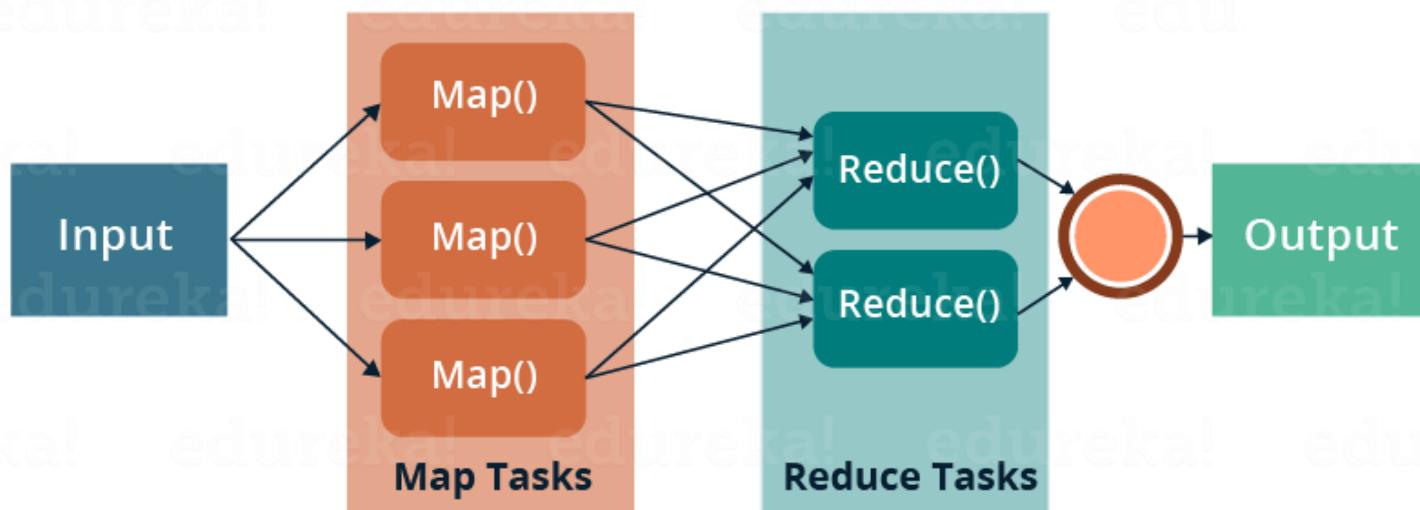
- Developer provides Mapper and Reducer functions
  - Developer decides what is key and what is value
  - Developer must follow the key-value pair interface
- 
- **Mappers:**
    - Consume <key, value> pairs
    - Produce <key, value> pairs
- 
- **Shuffling and Sorting:**
    - Groups all similar keys from all mappers,
    - sorts and passes them to a certain reducer
    - in the form of <key, <list of values>>
- 
- **Reducers:**
    - Consume <key, <list of values>>
    - Produce <key, value>



# What is MapReduce?

# What is MapReduce?

MapReduce is a **programming framework** that allows us to perform **distributed** and **parallel** processing on large data sets in a distributed environment

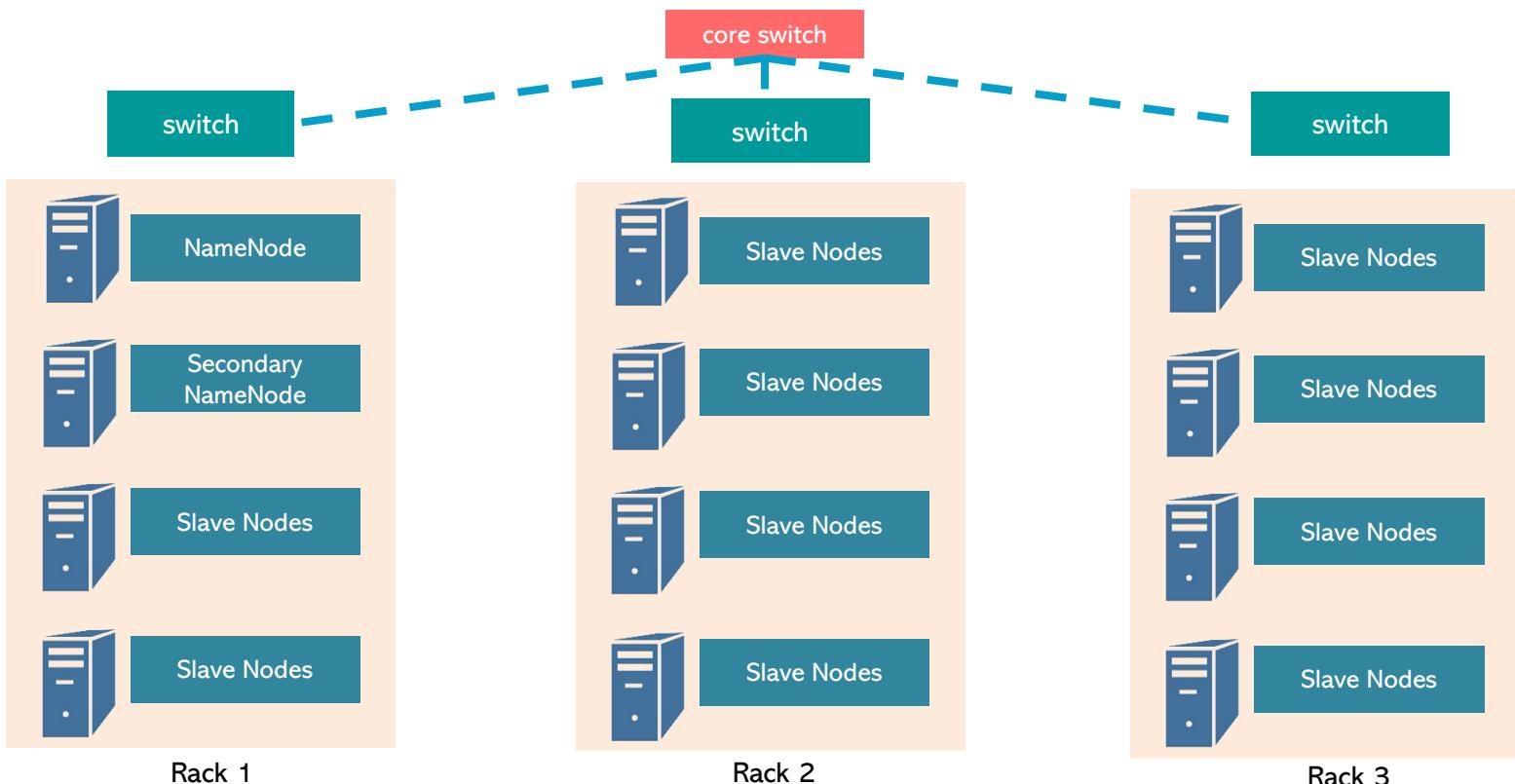




12

# Hadoop Cluster

# Hadoop Cluster



# Hadoop Cluster Nodes

# Hadoop Cluster Nodes

## Standalone (or Local) Mode

- No daemons, everything runs in a single JVM
- Suitable for running MapReduce programs during development
- Has no DFS or Distributed File System

## Pseudo Distributed Mode

- All Hadoop daemons run on the local machine

## Multi-Node Cluster Mode

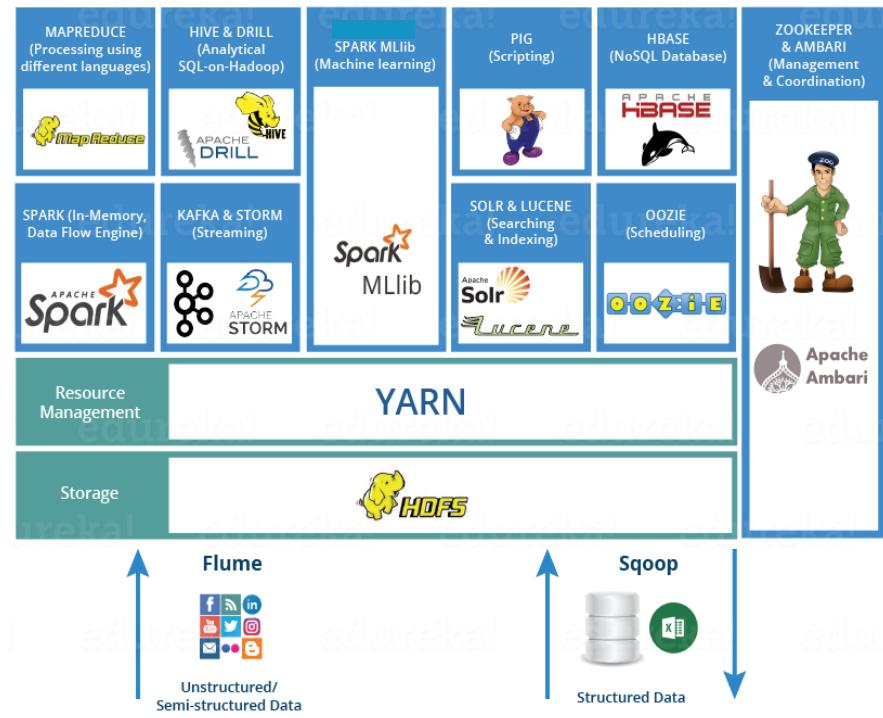
- Hadoop daemons run on a cluster of machines



13

# Hadoop Ecosystem

# Hadoop Ecosystem





14

Hands-On

# Hadoop Installation Steps

- Install Hadoop
- Step 1: Click here to download the Java 8 Package. Save this file in your home directory.
- Step 2: Extract the Java Tar File.
- Command: `tar -xvf jdk-8u101-linux-i586.tar.gz`
- Step 3: Download the Hadoop 2.7.3 Package.
- Command: `wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz`
- Step 4: Extract the Hadoop tar File.
- Command: `tar -xvf hadoop-2.7.3.tar.gz`
- Step 5: Add the Hadoop and Java paths in the bash file (`.bashrc`).
- Open `.bashrc` file. Now, add Hadoop and Java Path as shown below.
  
- Command: `vi .bashrc`
- Then, save the bash file and close it.
  
- For applying all these changes to the current Terminal, execute the source command.
  
- Command: `source .bashrc`
  
- To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the `java -version` and `hadoop version` commands.
  
- Command: `java -version`

# Hadoop Installation Steps

- To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.
- Command: java -version
- Command: hadoop version
- Step 6: Edit the Hadoop Configuration files.
- Command: cd hadoop-2.7.3/etc/hadoop/
- Command: ls
- All the Hadoop configuration files are located in hadoop-2.7.3/etc/hadoop directory as you can see in the snapshot below:
- Step 7: Open core-site.xml and edit the property mentioned below inside configuration tag:
- core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.
- Command: vi core-site.xml
- Editing Core-site - Install Hadoop - Edureka

# Hadoop Installation Steps

- Step 8: Edit hdfs-site.xml and edit the property mentioned below inside configuration tag:  
• hdfs-site.xml contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.
  - Command: vi hdfs-site.xml
- Step 9: Edit the mapred-site.xml file and edit the property mentioned below inside configuration tag:  
• mapred-site.xml contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.
  - In some cases, mapred-site.xml file is not available. So, we have to create the mapred-site.xml file using mapred-site.xml template.
  - Command: cp mapred-site.xml.template mapred-site.xml
    - Command: vi mapred-site.xml.
- Step 10: Edit yarn-site.xml and edit the property mentioned below inside configuration tag:  
• yarn-site.xml contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.
  - Command: vi yarn-site.xml

# Hadoop Installation Steps

- Formatting Name node
- After finishing the configuration, let's try to format the name node using the following command:
  - hdfs namenode -format
- Starting Hadoop services
- Now, we will open PowerShell, and navigate to "%HADOOP\_HOME%\sbin" directory. Then we will run the following command to start the Hadoop nodes:
  - .\start-dfs.cmd
- Next, we must start the Hadoop Yarn service using the following command:
  - ./start-yarn.cmd
- To make sure that all services started successfully, we can run the following command:
  - jps
- Hadoop Web UI
- There are three web user interfaces to be used:
  - Name node web page: <http://localhost:9870/dfshealth.html>
  - Data node web page: <http://localhost:9864/datanode.html>
  - Yarn web page: <http://localhost:8088/cluster>

# Multitenancy – What is it?

---



# Pros and Cons

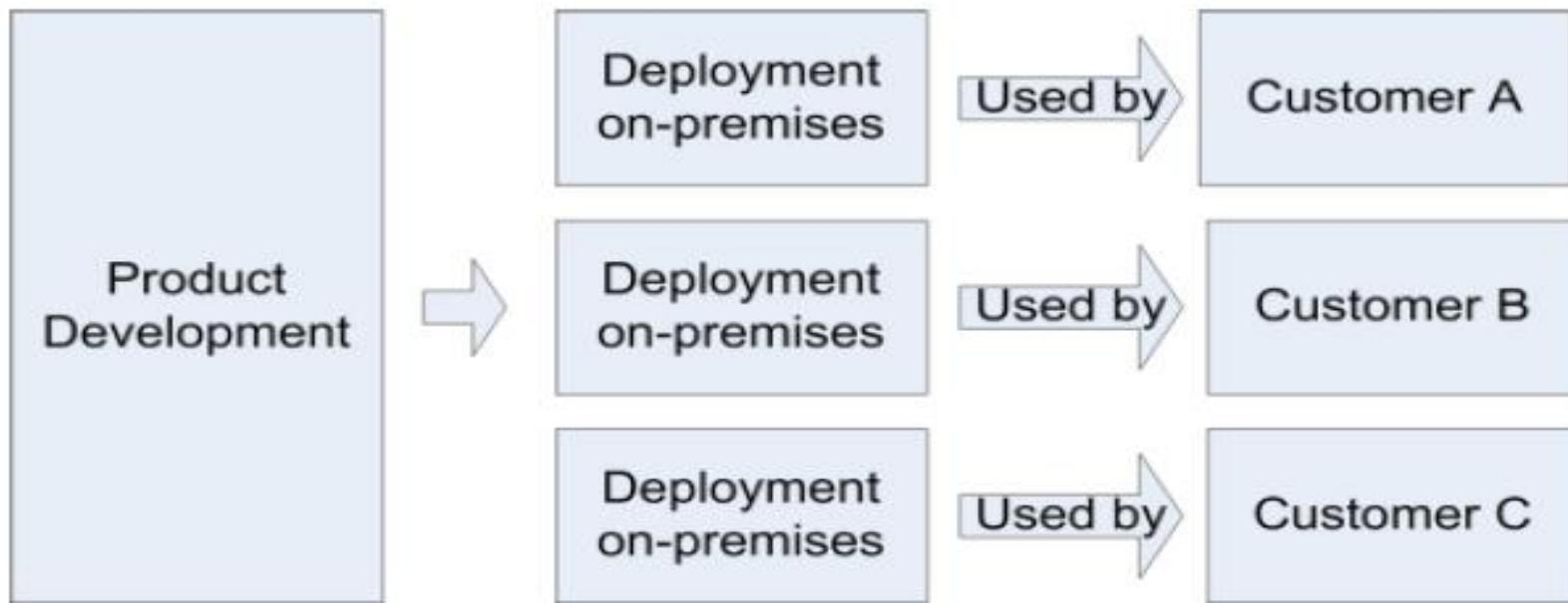
	House	Apartment
<i>Effective use of land</i>	-	+
<i>Privacy</i>	+	-
<i>Infrastructure sharing</i>	-	+
<i>Maintenance cost sharing</i>	-	+
<i>Freedom</i>	+	-

**House:** Privacy and freedom

**Apartment:** Cost efficiency

# Traditional Deployment Model

---



# Multitenancy – Introduction

---

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

# Multitenancy – Introduction

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

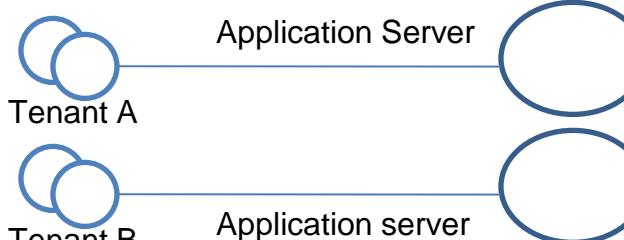
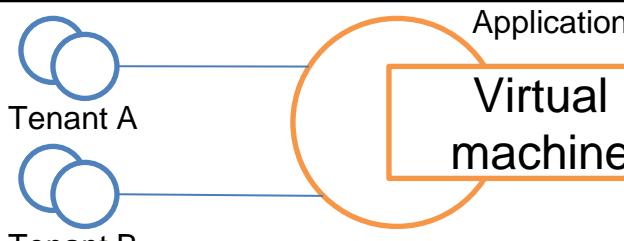
# Multitenancy – key aspects

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

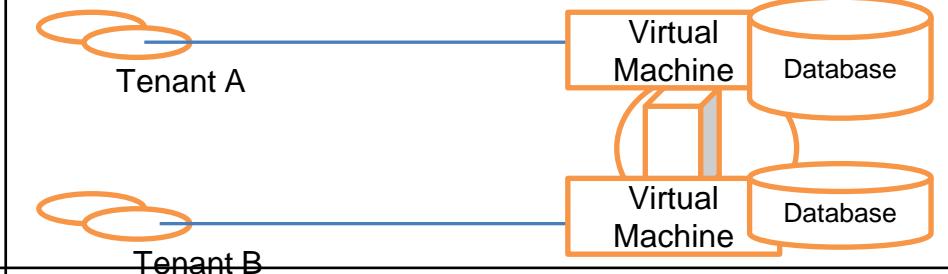
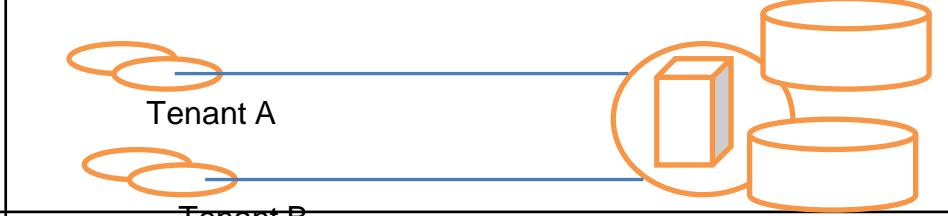
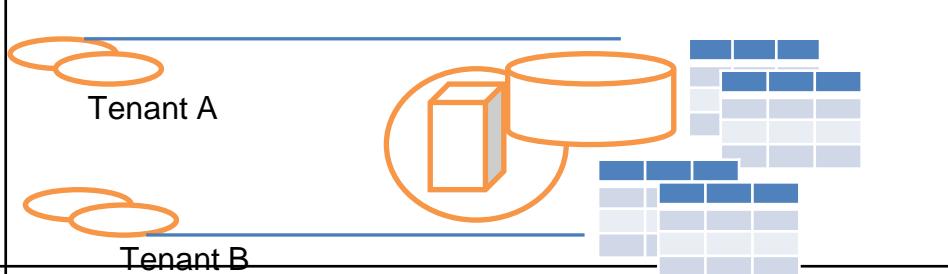
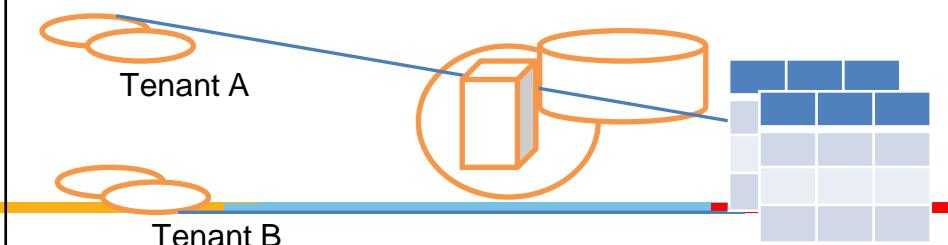
These definition focus on what we believe to be the key aspects of multi tenancy:

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

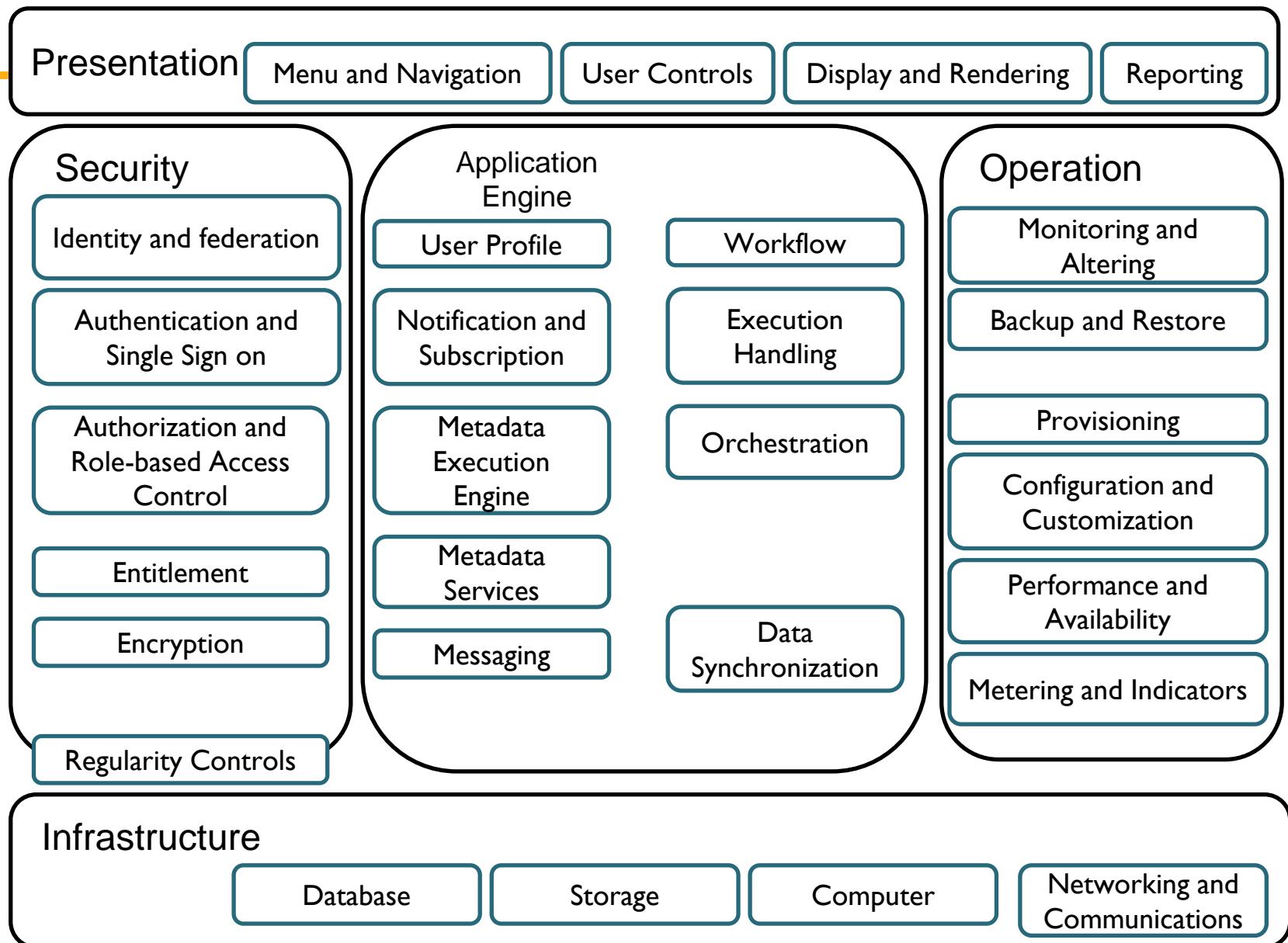
# Multi-tenants Deployment Modes for Application Server

<b>Fully isolated Application server</b> Each tenant accesses an application server running on a dedicated servers.	
<b>Virtualized Application Server</b> Each tenant accesses a dedicated application running on a separate virtual machine.	
<b>Shared Virtual Server</b> Each tenant accesses a dedicated application server running on a shared virtual machine.	
<b>Shared Application Server</b> The tenant shared the application server and access application resources through separate session or threads.	

# Multi-tenants Deployment Modes in Data Centers

<b>Fully isolated data center</b> <b>The tenants do not share any data center resources</b>	
<b>Virtualized servers</b> <b>The tenants share the same host but access different databases running on separate virtual machines</b>	
<b>Shared Server</b> <b>The tenants share the same server (Hostname or IP) but access different databases</b>	
<b>Shared Database</b> <b>The tenants share the same server and database (shared or different ports) but access different schema(tables)</b>	
<b>Shared Schema</b> <b>The tenants share the same server, database and schema (tables). The irrespective data is segregated by key and rows.</b>	

# Conceptual framework of Software as a Service





# Thank you

# Introduction to cloud security

---

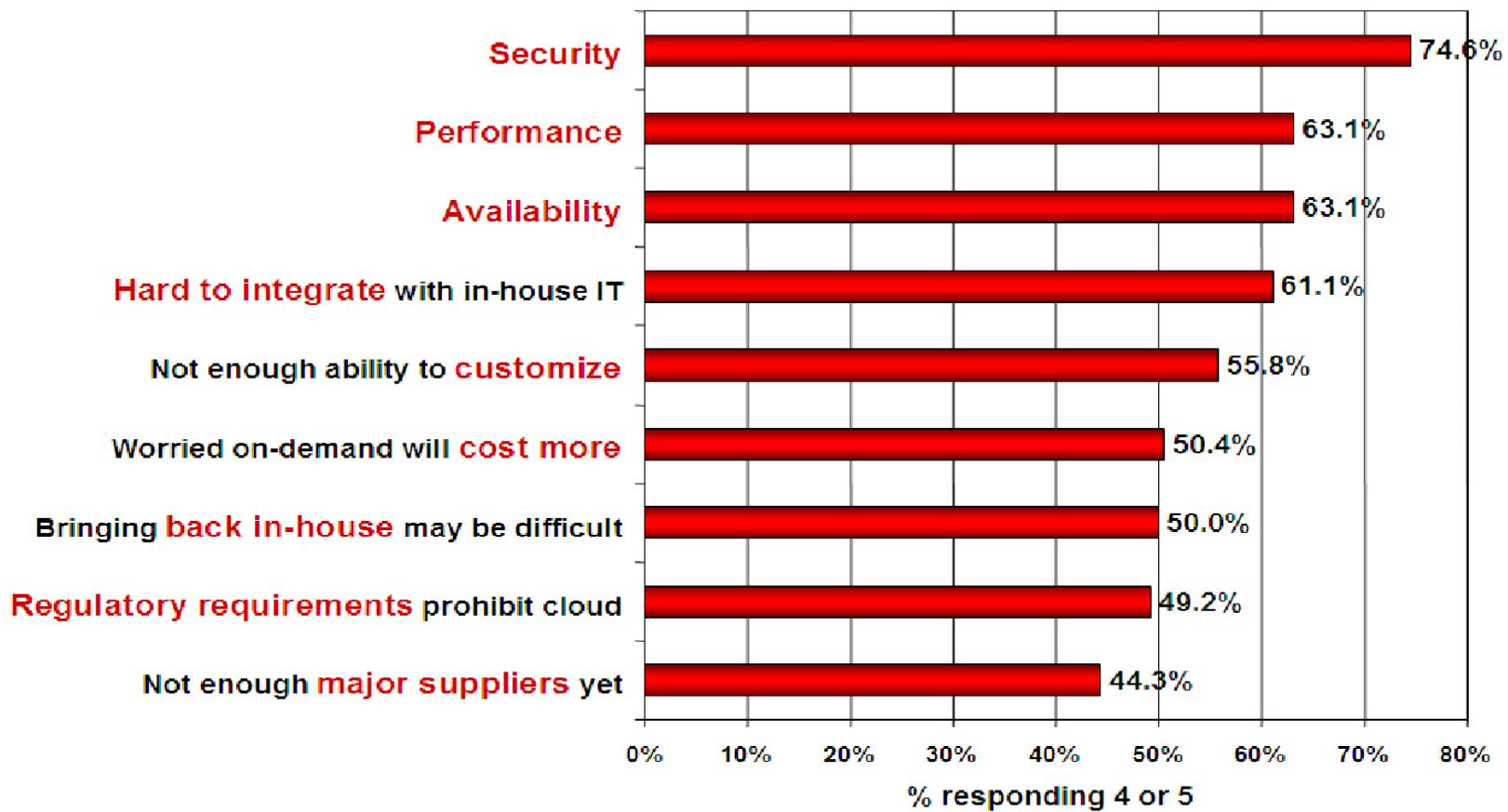
**If cloud computing is so great, why isn't everyone doing it?**

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

# Companies are still afraid to use clouds

Q: Rate the challenges/issues ascribed to the 'cloud'/on-demand model

(1=not significant, 5=very significant)



Source: IDC Enterprise Panel, August 2008 n=244

# Cloud Security Issues

---

- Most security problems stem from:
  - Loss of Control
    - Take back control
      - Data and apps may still need to be on the cloud
      - But can they be managed in some way by the consumer?
  - Lack of trust
    - Increase trust (mechanisms)
      - Technology
      - Policy, regulation
      - Contracts (incentives): topic of a future talk
  - Multi-tenancy
    - Private cloud
      - Takes away the reasons to use a cloud in the first place
    - VPC: it's still not a separate system
    - Strong separation
- These problems exist mainly in 3<sup>rd</sup> party management models
  - Self-managed clouds still have security issues, but not related to above

# Loss of Control in the Cloud

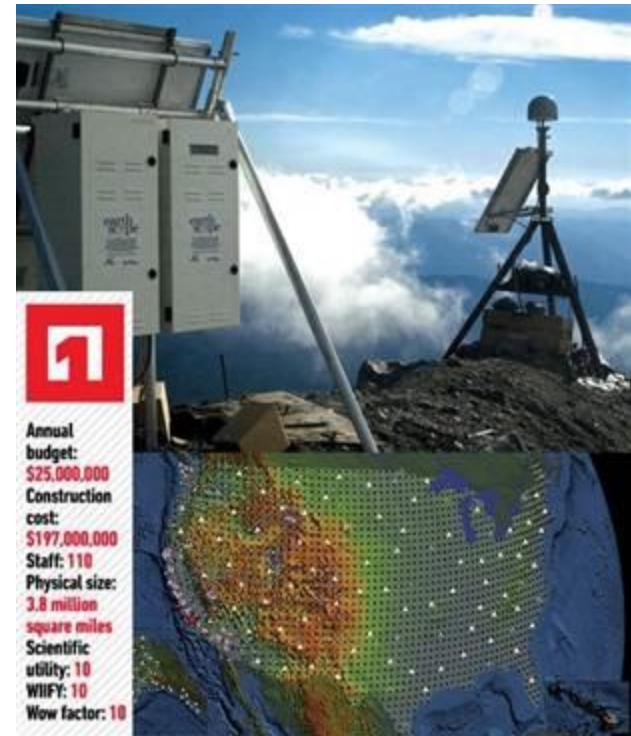
---

## Consumer's loss of control

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

# The Earthscope

- The Earthscope is the world's largest science project. Designed to track North America's geological evolution, this observatory records data over 3.8 million square miles, amassing 67 terabytes of data. It analyzes seismic slips in the San Andreas fault, sure, but also the plume of magma underneath Yellowstone and much, much more.  
[http://www.msnbc.msn.com/id/44363598/ns/technology\\_and\\_science-future\\_of\\_technology/#.TmetOdQ--ul](http://www.msnbc.msn.com/id/44363598/ns/technology_and_science-future_of_technology/#.TmetOdQ--ul))



# Multi-tenancy Issues in the Cloud

---

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

Multiple independent users share the same physical infrastructure

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

# Taxonomy of Fear

---

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

# Taxonomy of Fear

---

## Availability

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

# Taxonomy of Fear

---

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

# Taxonomy of Fear

---

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

# Threat Model

---

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

# Threat Model

---

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

## Cloud Fair Queuing

---

The algorithm ensures that a high-data-rate flow cannot use more than its fair share of the link capacity. Packets are first classified into flows by the system and then assigned to a queue dedicated to the flow. Packet queues are serviced one packet at a time in round-robin (RR) order.

[Interconnection networks](#) allow cloud servers to communicate with one another and with users. These networks consist of communication links of limited bandwidth and switches/routers/gateways of limited capacity. When the load exceeds its capacity, a switch starts [dropping packets](#) because it has limited input buffers for the switching fabric and for the outgoing links, as well as limited CPU cycles.

a [scheduling algorithm](#) has to manage several quantities at the same time: the *bandwidth*, the amount of data each flow is allowed to transport; the *timing* when the packets of individual flows are transmitted; and the *buffer space* allocated to each flow.

A first strategy to avoid [network congestion](#) is to use a [FCFS](#) scheduling algorithm. The advantage of the FCFS algorithm is a simple management of the three quantities: bandwidth, timing, and buffer space. Nevertheless, the FCFS algorithm does not guarantee fairness; greedy flow sources can transmit at a higher rate and benefit from a larger share of the bandwidth.

To address this problem, a [fair queuing](#) algorithm proposed in requires that separate queues, one per flow, be maintained by a switch and that the queues be serviced in a round-robin manner. This algorithm guarantees the fairness of buffer space management, but does not guarantee fairness of [bandwidth allocation](#). Indeed, a flow transporting large packets will benefit from a larger bandwidth

Quality of service (QoS) is the description or measurement of the overall performance of a service, such as a telephony or computer network, or a cloud computing service, particularly the performance seen by the users of the network.

System supports three types of QoS: application QoS, IP QoS and role QoS.

...

### **Introduction to QoS**

- Application QoS: Controls the bandwidth used by applications.
- IP QoS: Controls the bandwidth of designated IP addresses.
- Role QoS: Also called role-based QoS. It controls the bandwidth of designated roles.



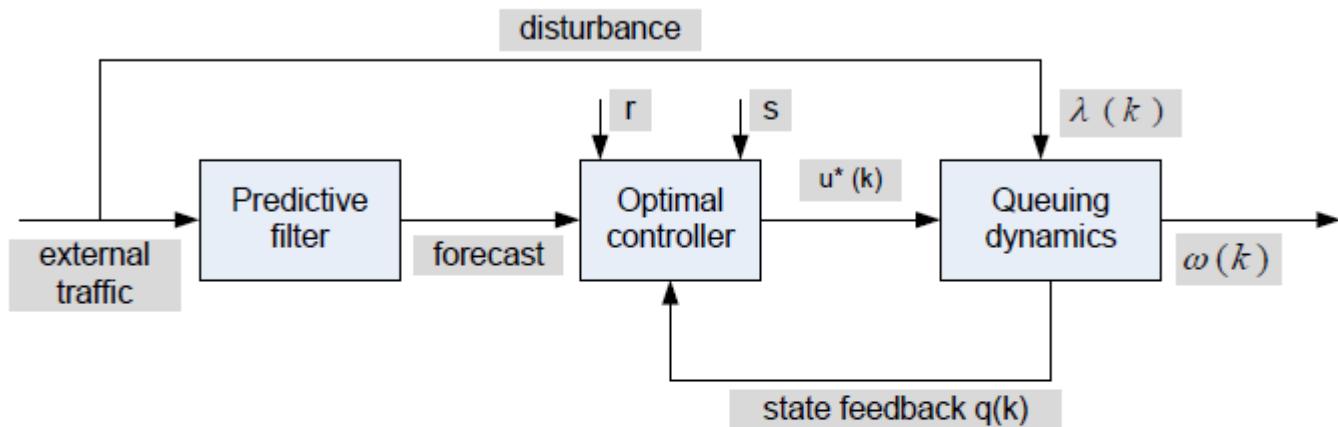
# Thank you

## Content

---

- Resource management and scheduling.
- Policies and mechanisms.
- Applications of control theory to cloud resource allocation.
- Stability of a two-level resource allocation architecture.
- Proportional thresholding.
- Coordinating power and performance management.
- A utility-based model for cloud-based Web services.
- Resource bundling and combinatorial auctions.
- Scheduling algorithms.
- Fair queuing.
- Start-up fair queuing.
- Borrowed virtual time.
- Cloud scheduling subject to deadlines.

## Cloud Control Structure



The controller uses the feedback regarding the current state and the estimation of the future disturbance due to environment to compute the optimal inputs over a finite horizon.  $r$  and  $s$  are the weighting factors of the performance index.

## Fundamental Questions

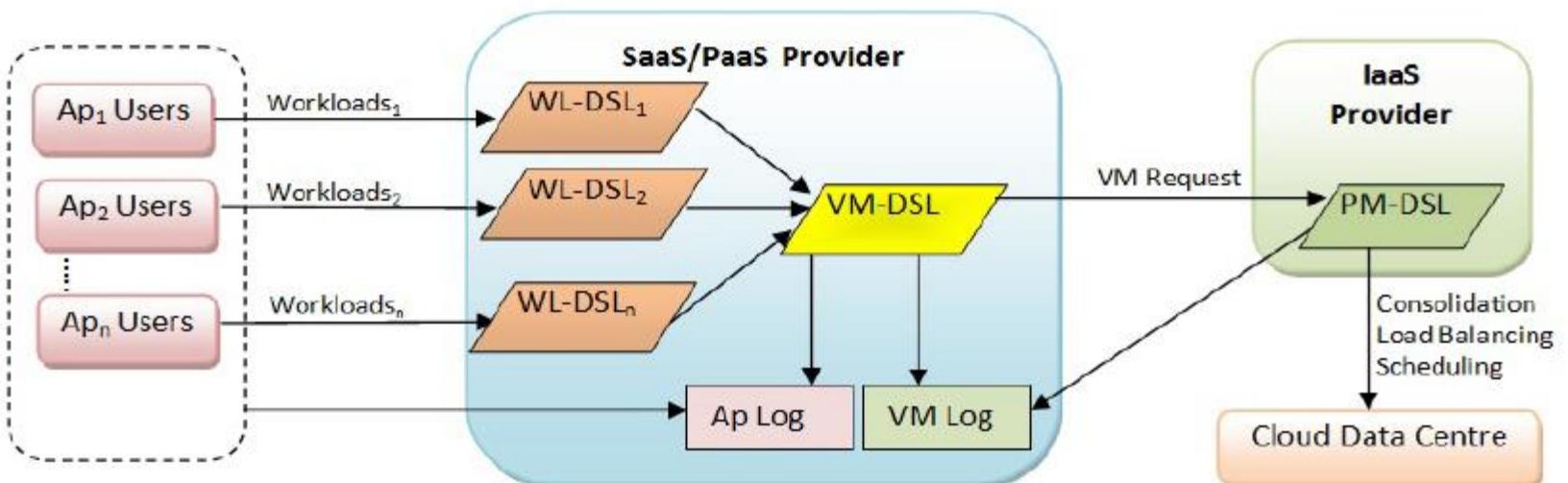
- What do I have?
- Which Cloud?
- What should I buy?
- How much should I buy? As provisioned or utilized?
- What are my buying options?
- What will it cost?
- How does it compare across clouds and on-premise?

## Work Load

<b>Application Specification</b>	<b>Workloads Generator</b>	<b>Workload Estimation</b>	<b>Parameter Access from Log File</b>
Web Application	Workload Specification Language (WSL) [2]. SPECweb99, SURGE, SWAT and httpref	Probabilistic Finite State Machine and Maximum Likelihood Estimation [2]	Input, output, states, transitions and probability of a transition
	Jean 2 model	Semantic descriptions [7]	<i>not applicable</i>
	KOOZA [5]	Markov Chain Models for storage, processor and memory. Simple queuing for network.	Storage: block size, type, randomness, inter-arrival times. Processor: CPU utilization. Network: arrival-rate
	<i>not applicable</i>	Autoregressive moving average method (ARMA) [13]	Number of visits to a single page from the total number of customers, number of machines providing the service demand and the think time for clients
Data Intensive	<i>not applicable</i>	Kernel Canonical Correlation Analysis [8]	Map time, reduce time, total execution time, map output bytes, HDFS bytes written, and locally written bytes
Media streaming	Medisyn [3]	Mathematical model in a tool called MediaProf	Time, file name, duration, file size, available users bandwidth and elapse end time

Different cloud platforms offer similar services with different characteristics, names, and functionalities. Therefore, describing cloud platform entities in such a way that they can be mapped to each other is critical to enable a smooth migration across platforms. DSL that uses a common cloud vocabulary for describing cloud entities covering a wide variety of cloud IaaS services.

# Work Load Capacity Management Framework



---

## **Why do we use HDFS for applications having large data sets and not when there are a lot of small files?**

HDFS is more suitable for large amounts of data sets in a single file as compared to small amount of data spread across multiple files. As you know, the NameNode stores the metadata information regarding the file system in the RAM. Therefore, the amount of memory produces a limit to the number of files in my HDFS file system. In other words, too many files will lead to the generation of too much metadata. And, storing these metadata in the RAM will become a challenge. As a thumb rule, metadata for a file, block or directory takes 150 bytes.

---

---

## **How do you define “Rack Awareness” in Hadoop?**

**Rack Awareness** is the algorithm in which the “NameNode” decides how blocks and their replicas are placed, based on rack definitions to minimize network traffic between “DataNodes” within the same rack. Let’s say we consider replication factor 3 (default), the policy is that “for every block of data, two copies will exist in one rack, third copy in a different rack”. This rule is known as the “Replica Placement Policy”.

## **What is the difference between an “HDFS Block” and an “Input Split”?**

The “HDFS Block” is the physical division of the data while “Input Split” is the logical division of the data. HDFS divides data in blocks for storing the blocks together, whereas for processing, MapReduce divides the data into the input split and assign it to mapper function.

---

---

**state the reason why we can't perform "aggregation" (addition) in mapper? Why do we need the "reducer" for this?**

This answer includes many points, so we will go through them sequentially.

- We cannot perform "aggregation" (addition) in mapper because sorting does not occur in the "mapper" function. Sorting occurs only on the reducer side and without sorting aggregation cannot be done.
- During "aggregation", we need the output of all the mapper functions which may not be possible to collect in the map phase as mappers may be running on the different machine where the data blocks are stored.
- And lastly, if we try to aggregate data at mapper, it requires communication between all mapper functions which may be running on different machines. So, it will consume high network bandwidth and can cause network bottlenecking.

---

## **How do “reducers” communicate with each other?**

This is a tricky question. The “MapReduce” programming model does not allow “reducers” to communicate with each other. “Reducers” run in isolation.

## **What does a “MapReduce Partitioner” do?**

A “MapReduce Partitioner” makes sure that all the values of a single key go to the same “reducer”, thus allowing even distribution of the map output over the “reducers”. It redirects the “mapper” output to the “reducer” by determining which “reducer” is responsible for the particular key.

## What is Apache HBase?

HBase is an open source, multidimensional, distributed, scalable and a NoSQL database written in Java. HBase runs on top of HDFS (Hadoop Distributed File System) and provides BigTable (Google) like capabilities to Hadoop. It is designed to provide a fault-tolerant way of storing the large collection of sparse data sets. HBase achieves high throughput and low latency by providing faster Read/Write Access on huge datasets.

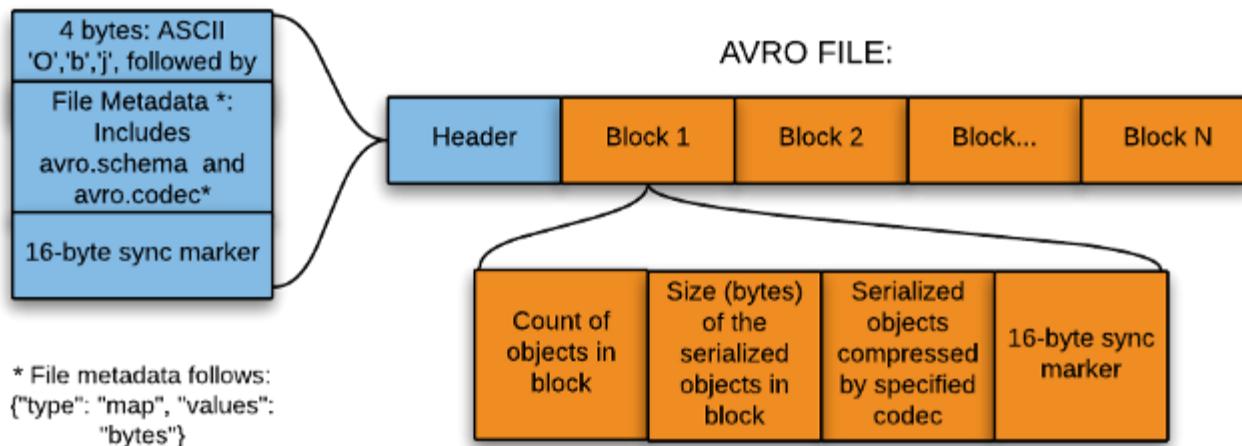
---

## 7. What are the different types of file formats used to store in the Apache Hadoop?

CSV, JSON, Columnar, Sequence files, AVRO and Parquet file are some of the files used in Apache Hadoop.

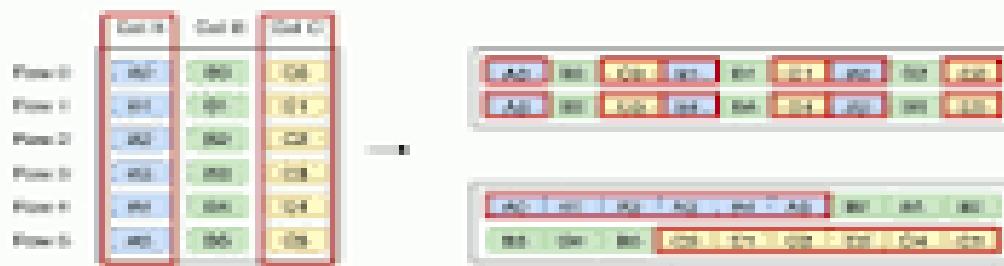
Columnar-he *columnar* data formats are a popular choice for fast analytics workloads. As opposed to row-oriented storage, columnar storage can significantly reduce the amount of data fetched from disk by allowing access to only the columns that are relevant for the particular query or workload.

Avro-Avro format is a row-based storage format for **Hadoop**, which is widely used as a serialization platform. Avro format stores the schema in JSON format, making it easy to read and interpret by any program. The data itself is stored in a binary format making it compact and efficient in Avro files

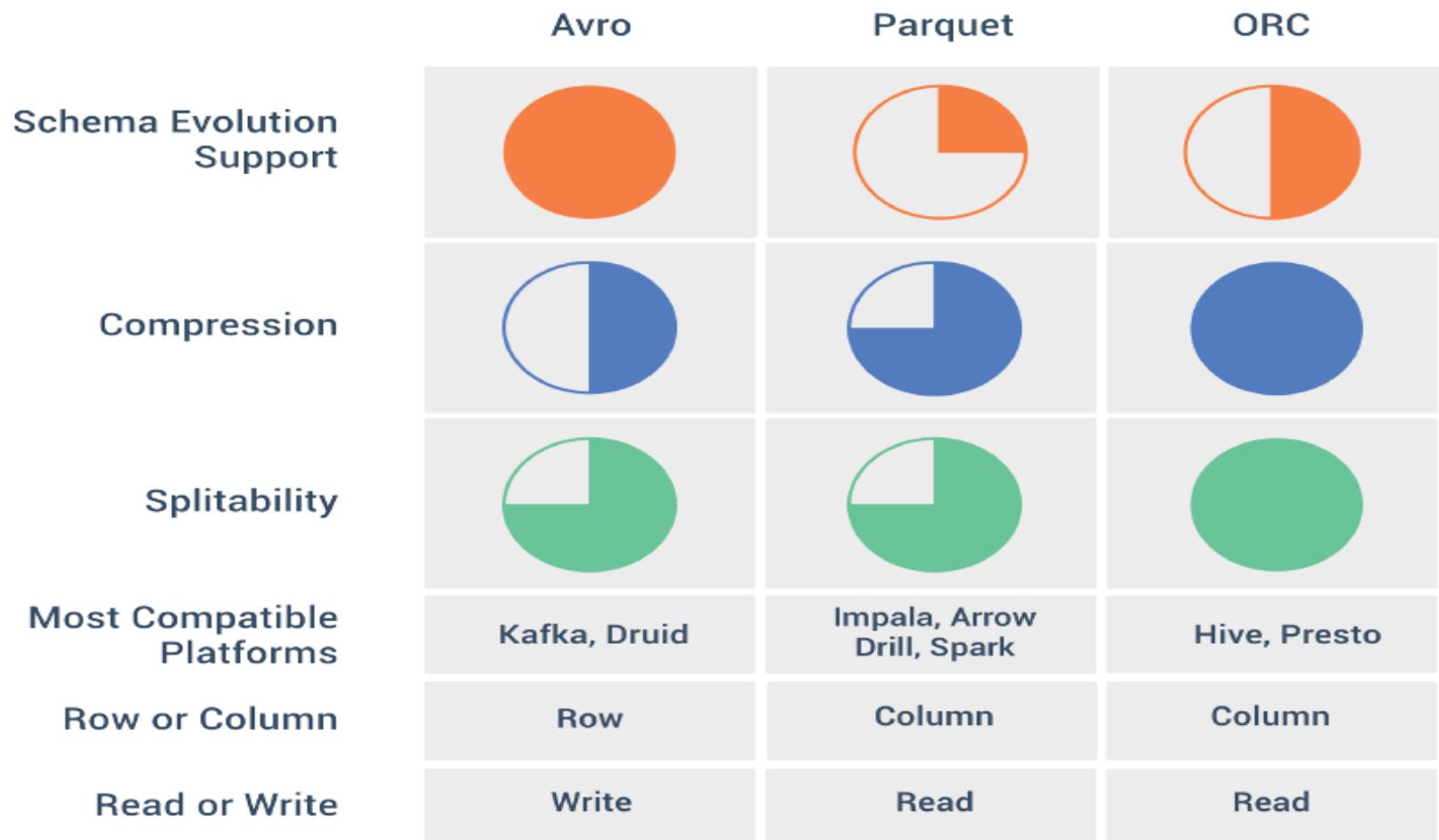


Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

## Row-wise vs Columnar



## BIG DATA FORMATS COMPARISON



Source: Nexla analysis, April 2018

# MapReduce Job Workflow

