

Cloud Computing SEZG527

CLOUD COMPUTING	1
Session Agenda	1
Cloud Skills	2
Module 1 - Introduction to Cloud Computing	2
What is Cloud Computing Your Opinion ?	3
Why is it Called Cloud	3
Whats in the Cloud ??	4
Familiar Use Cases	4
Facts about Cloud	5
Facts about Cloud - Now	5
Cloud Origins & Motivation	6
Origins	6
Paradigms	7
How do we define Cloud ?	8
Terms to Remember	8
Cloud NIST & the AAS es	9
NIST 3-4-5 Rule for Cloud	9
Cloud Service Models (AAS es)	10
Infrastructure as a Service	10
Platform as a Service	12
Software as a Service	13
Which AAS?	14
Who Manages the AAS es?	15
Cloud Deployment Models	15
Deployment Models in Cloud	16
Pubic Cloud	16
Private Cloud	18
Hybrid Cloud	19
Community Cloud	21
Quick Comparison	23
Cloud Essential Characteristics	23
Essential Characteristics	24
Resource pooling	24
Broad Network Access	25
On Demand Self Service	25
Rapid Elasticity	26
Metered by Use	26

Cloud Advantages	27
Cloud vs Hosted	27
Cloud Challenges	28
Cloud ecosystem	29
Cloud Failures	30
NIST Definitions	31
Module 2.- Virtualization Techniques and Types	32
Virtualization History	32
Motivations & Origins	33
Video Virtualization	33
What is Virtualization?	34
Virtualization Architecture	35
Hypervisor	36
Hypervisor Goals	36
Hypervisor - Samples	37
Video BOCHS	37
Hypervisor Types	38
Hypervisor Architecture	38
Comparison	39
Resource Sharing in VM - CPU	40
Resource Sharing in VM - Memory	41
Resource Sharing in VM - IO	41
Hypervisor Techniques	42
Benefits of Virtualization	43
Virtualization Summary	44
Key Terms to Remember	44
What is Virtualization?	45
Understanding Hypervisor	46
What is Hypervisor	46
Why use Hypervisor	47
Type 1 Hypervisor	47
Type 1 Hypervisor Pros	48
Type 1 Hypervisor Cons	48
Type 2 Hypervisor	49
Type 2 Hypervisor - PROs	49
Type 2 Hypervisor - CONs	50
Type 1 & 2 Hypervisor At a Glance	50
Virtualization Evolution	51
x86 Architecture	51
Challenges to Virtualization	52
VMware's Solution	53

Virtualization Evolution	53
Introduction	54
Virtualization Approaches	54
Emulation	55
Binary Translation / Full Virtualization	56
Binary Translation / Full Virtualization - Pros	56
Binary Translation / Full Virtualization - Cons	57
Para Virtualization	57
H/W Assisted Virtualization	58
SKI Virtualization	59
Virtualization Comparison	60
Virtualization Types	60
Virtualization	61
Server Virtualization	61
Server Virtualization - Benefits	62
Storage Virtualization	62
Storage Virtualization - Benefits	63
Network Virtualization	63
Memory Virtualization	64
Device Virtualization	65
Virtualization Advantages	65
Virtualization Summary	66
Virtualization Advantages	66
Points to Note	68
Application of Virtualization	69
Technology Trends	69
What is Virtualization?	70
Virtualization Comparison	71
Virtualization	71
Module 3 - Infrastructure as a Service	72
What is IaaS?	72
Infrastructure as a Service	73
Visualization for a New Paradigm	73
IaaS - Motivations	74
IaaS Key Terms	74
Pros & Cons of IaaS	75
Introducing Amazon Web Service	75
What is AWS	76
AWS Understanding Service Offering	76
AWS Global Infrastructure	77
AWS Regions	77

AWS Availability Zones	78
AWS Local Zones	78
AWS Wavelength Zones	79
AWS Outposts	79
AWS for the Edge	80
AWS & Customer	80
Using AWS - Connecting	81
Using AWS - Video	81
AWS Reference Model	82
AWS Elastic Compute Cloud EC2	82
EC2 Introduction	83
EC2 - Instance Type & AMI	83
EC2 - Instance Type	84
EC2 - Amazon Machine Image (AMI)	85
EC2 - AMI Types	85
Connecting to EC2 Instance	87
Creating an EC2 Instance - Video	88
EC2 Accessing over Web	88
EC2 Lifecycle	89
EC2 Tenancy Options	89
EC2 Placement Groups	90
Networking in AWS - VPC	90
AWS VPC	91
AWS VPC - Components	91
AWS VPC - Functioning	92
AWS Storage	92
Storage Types	93
AWS Simple Storage Service S3	93
AWS S3	94
AWS Elastic Block Storage EBS	95
AWS EBS	96
AWS EBS Types	96
AWS Elastic File Storage EFS	97
AWS EFS	98
AWS EFS vs AWS EBS vs AWS S3	98
AWS Glacier	99
Amazon Glacier	99
AWS Case Study	100
Netflix	100
Agenda	101
What is AWS	102

AWS Regions & Availability Zones	102
AWS Reference Model	103
OpenStack	103
What is OpenStack?	104
History of Open stack	104
Mission Statement	105
What you get with OS	105
OpenStack Projects	106
OpenStack Reference Model	107
OpenStack Projects	107
OpenStack Services	108
Introducing Network Functions Virtualization - NFV	109
What is NFV?	109
NFV Origins & Motivations	110
What is NFV	111
NFV Architecture	112
NFV Comparison	112
NFV Benefits	113
Comparing AWS & OpenStack	113
Overview	114
Differences	114
OpenStack Case Study	115
Generic 3-tier Web Application	116
OpenStack Services Used	116
What is AWS	117
AWS Regions & Availability Zones	118
AWS Reference Model	118
What is OpenStack?	119
OpenStack Reference Model	119
Differences	120
VM Management	121
Anatomy of Cloud	121
What do we manage in Public Cloud?	122
What do we manage in Private Cloud?	123
Why Resource Management	123
What is Resource Management	124
VM Lifecycle	125
VM Provisioning process	126
VM Provisioning using templates	127
Understanding Vagrant	127
Demo HEAT Provisioning	128

VM Migration	128
What is VM Migration	129
What is VM Live Migration	130
Live Migration Xen Hypervisor	130
Live Migration Process	132
Live Migration Technique	133
Live Migration Effect on a Running Web Server	134
Live Migration Vendor Implementations Example	134
Cold Migration	135
Storage Migration	135
FUTURE DIRECTIONS	136
Virtualization	137
Module 4 - Containers	137
Motivation towards Containers	138
What are Containers?	138
Linux Containers	139
Linux Containers LXC	139
Control Groups	140
Linux Containers LXD	141
Comparing LXC & LXD	142
Container Types OS Containers	142
Minimalistic OS	143
Container Types Application Containers	143
Comparison OS vs Application Containers	144
Difference between VM & Container	144
Dockers	145
Dockers - Motivation	145
Dockers - Introduction	146
Dockers - Architecture	147
Dockers Components - Daemon	147
Dockers Components Client	148
Dockers Components Registries	148
Docker Objects	149
Docker Objects - Images	149
Docker Objects - Containers	151
Docker Objects - DockerFiles	152
Docker Life Cycle	153
Docker Summary	153
Container Orchestration	154
IAC	155
Imperative vs Declarative Model in IAC	156

Container Orchestration	157
Kubernetes Introduction Video	158
Container Orchestration - Tools	158
Motivation towards Containers	159
What are Containers?	160
Dockers - Motivation	160
Difference between VM & Container	161
Module 5 - Platform as a Service and Software as a Service (PaaS & SaaS)	161
Introducing Platform as a Service	162
Building Blocks - PaaS	162
Characteristics- PaaS	163
PaaS vs IaaS	163
PaaS Advantages	164
PaaS Disadvantages	164
Who Can Use PaaS	165
PaaS Best Practices	165
PaaS AWS Examples	166
Platform as a Service - Summary	166
Windows Azure	167
What is Microsoft Azure?	167
Azure Components?	168
Azure Services?	168
Azure key Terms?	169
Azure PaaS Design Principles	170
Azure Introduction	170
Azure Runtime Environment	171
Azure fabric Controller	171
Azure Components	172
PaaS Vendors (Popular)	172
Software as a Service	173
Agenda	173
Introducing Software as a Service	174
SaaS Motivations	174
SaaS Delivery Model	175
SaaS Architecture	175
SaaS Model Comparison	176
SaaS Adantages	178
SaaS Providers	178
SaaS Providers at a Glance	179
SaaS User Benefits	180
SaaS Vendor Benefits	181

Software as a Service	182
SaaS Applicability Scenarios 1 2 3	183
The Right SaaS Model?	183
SaaS vs PaaS vs IaaS COMPARISON	184
SaaS vs PaaS vs IaaS APPLICABILITY	184
Security in the Cloud	185
Objective	186
Analysis of Security Need	186
Cloud Governance (L to R)	187
Why Reluctance ?	187
What the Future Holds?	188
Cloud Security Issues	188
Loss of Control Trust Issues	189
MT Issues in Cloud	189
Taxonomy of Fear	190
Threat Model	191
Threat Sources	191
Data Security and Storage	192
What is Privacy?	192
Security in the Cloud Possible Solutions	193
Security Issues in the Cloud The What	193
Security Issues in the Cloud My Diagnosis CLOUD SECURITY ISSUES DETECTION DOES NOT MINIMIZE THE RISK, YOU NEED TO REMEDY.	194
Security Issues in the Cloud Remedy	194
Minimize Lack of Trust: Policy Language	196
Minimize Lack of Trust: Certification	196
Minimize Loss of Control: Monitoring	197
Minimize Loss of Control: Access Control	198
Minimize Multi-tenancy	199
Conclusion	199
Agenda	200
Software as a Service - Summary	200
True SaaS Applicability	201
SaaS vs PaaS vs IaaS COMPARISON	201
Module 6 Capacity Management and Scheduling in cloud computing	202
Anatomy of a Cloud Ecosystem	202
Why Capacity Management in Cloud?	203
What are the Challenges?	203
Importance of Capacity Management	204
So Whats the Way? Find the Right Balance	204
How To ? Find the Right Balance	205

What is Virtual Infrastructure Management	206
Virtual Infrastructure Management	206
So What is the way out? VIM	207
Virtual Infrastructure Manager (VIM)	208
What is a Virtual Infrastructure Manager?	208
Why a Virtual Infrastructure Manager?	209
Enter Distributed Management of Virtual Resources	210
So What is the Solution	210
OpenNebula VIM	211
What is OpenNebula	211
Stages of VM Life Cycle in OpenNebula	212
VM Management in OpenNebula	212
Image Management in OpenNebula	213
Networking OpenNebula	213
Benefits of OpenNebula	214
Features of OpenNebula	214
Comparison with Similar Technologies	215
Scheduling VM Workloads	215
What is a Cloud Workload	216
What are we Talking about	217
Where do we Run Workloads?	217
Workload Challenges	218
Cloud Workload An Anatomy	219
Amdahls Law	219
Scheduling Techniques	220
Scheduling Techniques Existing Approaches	220
Scheduling Techniques VM Overheads	222
Reservation Based Provisioning	223
Provisioning to meet SLA	223
Scheduling Techniques for Advance Reservation of Capacity	224
Solution Haizea Scheduler	225
What is Haizea Scheduler	225
How Haizea Scheduler Works	226
Leasing Schedule Best Effort Lease (BEL)	228
Leasing Schedule Advanced Reservation (AR)	229
Capacity Management to meet SLA Commitments	230
Infrastructure SLA's	230
Anatomy of a Cloud Ecosystem	233
Importance of Capacity Management	234
What are the Challenges?	234
What is a Virtual Infrastructure Manager?	235

What is a Cloud Workload	235
Module 7 - Issues and Challenges: Availability, Multi-Tenancy, Security and SLA	236
Multi Tenancy	236
What is Multi Tenancy?	236
Some Facts	237
Multi Tenant vs Multi Instance	238
IAAS MT Model	239
PAAS MT Model	239
SAAS MT Model	240
Benefits	240
Disadvantages	241
Characteristics of MT Architecture	242
MT- Different Levels	242
Security in MT	244
Multi Tenancy: Resource Sharing	245
Resource Sharing Approaches	245
Support for Customization	246
Sample MT Architecture	248
Multi Tenancy: Resource Sharing	248
Did You Know ?	249
SEAY-ZG527 CLOUD COMPUTING	250
SECURITY	251
Introduction	251
Cloud Computing: Security Analysis?	252
Impact of cloud on the governance structure of IT organizations	252
Cloud Adoption : Reluctance	253
Companies are afraid to use clouds	253
Cloud Security Issues	254
Loss of Control in the Cloud	254
Lack of Trust in the Cloud	255
Multi-tenancy Issues in the Cloud	255
Taxonomy of Fear - CIA	256
Taxonomy of Fear (cont.)	256
Threat Model	257
What is the issue?	258
Attacker Capability: Malicious Insiders	259
Attacker Capability: Outside attacker	259
Challenges for the attacker	260
PART II: SECURITY AND PRIVACY ISSUES IN CLOUD COMPUTING - BIG PICTURE	260
Data Security and Storage	261
What is Privacy?	261

PART III. POSSIBLE SOLUTIONS	262
Security Issues in the Cloud	262
Minimize Lack of Trust: Policy Language	263
Minimize Lack of Trust: Certification	263
Minimize Loss of Control: Monitoring	264
Minimize Loss of Control: Monitoring (Cont.)	264
Minimize Loss of Control: Utilize Different Clouds	265
Minimize Loss of Control: Access Control	265
Minimize Multi-tenancy	266
Conclusion	266
What is Multi Tenancy?	267
Some Facts	268
Multi Tenant vs Multi Instance	269
MT- Different Levels	269
SLA Management	270
What is SLA or Service Level Agreement	271
SLA Role in High Availability	271
Steps for HA	272
The 3 Initialisms	273
SLA vs SLO	273
The 3 Initialisms	274
SLA	274
Co-Hosting Application	276
Co-Hosting Application: Issues	276
Co-Hosting Application: Solution	277
Traditional SLO Management Approaches	277
SLA Types	278
Infrastructure SLA vs Capacity Management	279
Infrastructure SLA	279
SLA Life Cycle	280
SLA LC Management Cloud Applications	282
SLA Cloud Application LC Management	283
Module 8 - DFS, GFS, MR and Hadoop	287
File System- Introduction	288
Distributed File System- Introduction - What is Distributed File System	288
Components of Distributed File System	289
Distributed File System Challenges	289
Google File System	290
GFS Assumptions	291
GFS is not suited	291
GFS Architecture	292

GFS Design Overview	292
Files on GFS	293
GFS - Master	293
GFS - Chunks	294
GFS - Metadata	294
GFS - Operational Logs	295
GFS - Chunk Replica	295
Big data	296
Data Evolution	296
What is Big data?	297
Hadoop	297
Hadoop – HDFS Design	298
HDFS BLOCKS	300
HDFS Name nodes and data nodes	301
HDFS REDUNDANCY (High Availability)	302
HDFS BLOCK CACHING	302
HDFS FEDERATION	303
Hadoop Distributed File System	303
HDFS File Read	304
HFDS YARN	304
MapReduce (Data Processing Framework)	305
HDFS MR	305
MapReduce Processing flow	306
Architecture Overview	306
Classes of problems mapreducible	307
Fault Tolerance in MapReduce	308
Some Facts about MR	309
Session Agenda	310
Cloud Deployments	311
Perception	311
BaaS Defined	312
BaaS Architecture	312
BaaS BaaS Examples	313
BaaS vs IaaS	313
BaaS Pro & Con	314
Serverless Computing	314
BaaS vs Serverless	315
Serverless on AWS	316
Serverless Advantages	317
Serverless Disadvantages	318
Function as a Service	318

FaaS	319
Microservice	319
FaaS Benefits	320
Mobile BaaS	320
MBaaS	321
MBaaS Architecture	321
Future Directions	322
What Next in Cloud	322
EDGE Computing	323
EDGE Computing Use Cases	324



BITS Pilani
Pilani Campus

CLOUD COMPUTING

Session 1

Arun Vadekkedhil



Session Agenda



Cloud Computing – An Overview

- Introductions – Tutor, Course & Students
- Paradigms & Distributed Computing
- origins & Motivation for Cloud
- What is Cloud Computing
- Is Cloud Computing for me?

Types of Clouds & Service Deployment

- 3-4-5 Rule of Cloud Computing
- Cloud Infrastructure & Deployment

Wrap Up

- Perceived benefits of Cloud ecosystem
- Challenges to Overcome
- Advantages & Disadvantages
- Commercial offering of Cloud services
- Reality Check

Cloud Skills

Cloud engineer skills at a glance



BITS Pilani, Pilani Campus



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



Cloud – An Introduction



What is Cloud Computing



Your Opinion?



BITS Pilani, Pilani Campus

Why is it Called Cloud

Cloud computing, eponymously is named after the cloud symbol used in architecture documents.

By now, you must be aware that this has no relation to its namesake from the meteorology department, but it simply means that we are using the internet to store data on remote servers, rather than storing them locally on our hard disks.

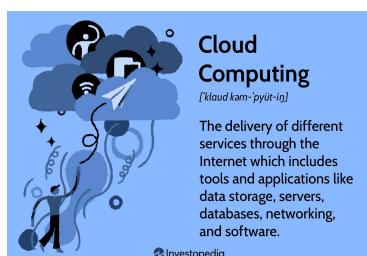
Many types of Cloud Computing Applications Exist. Depending on the need of your IT the type of cloud solution may vary.

Cloud Computing is up to 40 times more cost-effective for an SMB compared to running its own IT system or department.

Cloud Computing

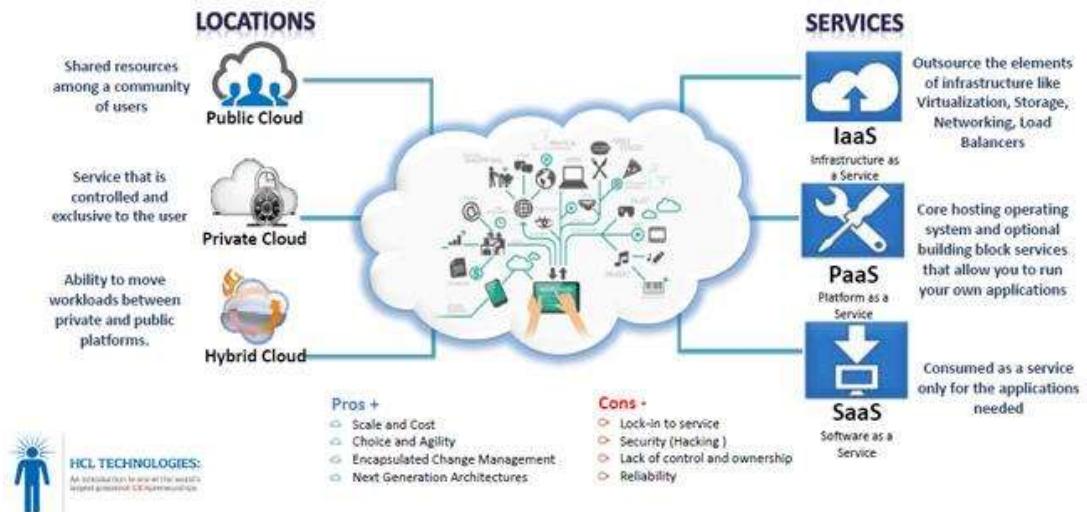


Why is it called the cloud?



What's in the Cloud ??

What is Cloud Computing?

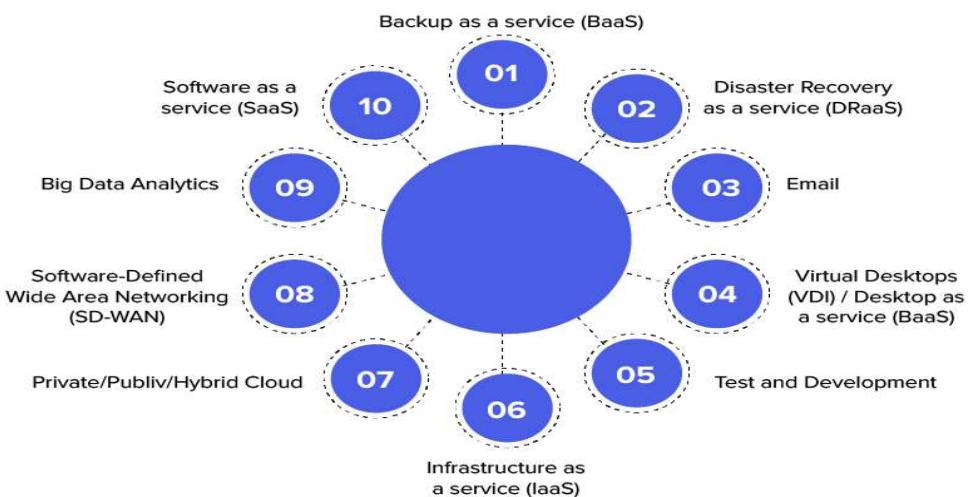


[Image Credit](#)

BITS Pilani, Pilani Campus

Familiar Use Cases

Cloud Computing Use Cases



[Image Credit](#)

BITS Pilani, Pilani Campus



Facts about Cloud

Here are our top cloud facts for 2022:

1. By 2022, more than 90% of enterprises will rely on a hybrid cloud environment to meet their infrastructure needs. ([Source](#))
2. The value of the cloud computing market is estimated to be \$832.1 billion by the end of 2025 (compared to \$371.4 billion in 2020). ([Source](#))
3. Cloud security is a top concern for 75% of enterprises. ([Source](#))
4. By 2025 there will be over 100 Zettabytes of data stored in the cloud (1 zettabyte = 1 billion terabytes = 1 trillion gigabytes). ([Source](#))
5. 50% of companies reported higher cloud usage than planned during the Covid-19 pandemic. ([Source](#))
6. 61% of organizations want to optimize cloud spend, making it the top initiative for the 5th year in a row. ([Source](#))
7. The public cloud sector is expected to generate \$331 billion in revenue by 2022 (up from \$175.8 billion in 2018). ([Source](#))
8. Spending on IT infrastructure is predicted to reach \$55.7 billion by 2022. IDC predicts a 10.9% growth in demand for servers, switchers, and storage solutions.
9. Platform-as-a-Service (PaaS) grew in adoption to 56% in 2021, making it the fastest growing segment in cloud platforms. ([Source](#))
10. 93% of businesses have a multi-cloud strategy. As these deployments mature, cost containment and cybersecurity will be top priorities. ([Source](#))

BITS Pilani, Pilani Campus



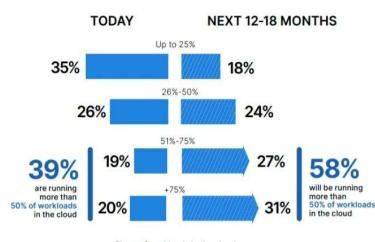
Facts about Cloud - Now

More than 90% of organizations use the cloud (Source: O'Reilly)

O'Reilly's latest Cloud Adoption report had some interesting numbers. Consider this:

- About two-thirds of respondents currently operate in a public cloud and 45% use a private cloud – versus 55% who still rely on traditionally managed on-premises systems.
- 48% plan to migrate at least half of their applications to the cloud in the next year; 20% intend to move all their applications to the cloud.
- 47% are pursuing a cloud-first strategy; 30% are already cloud-native; 37% intend to be cloud-native in about three years.
- Only 5% plan on switching from the cloud to on-premises infrastructure (cloud repatriation).

- What percentage of your workloads are in the cloud today? ► What percentage of your workloads will be in the cloud in the next 12-18 months?



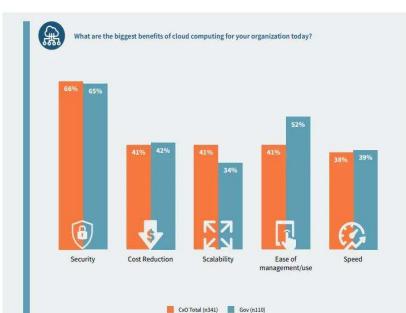
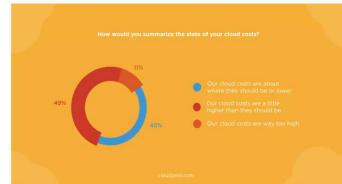
6 in 10 CxOs say cloud computing improves security (Source: Oracle)

Security is the top benefit of cloud computing, according to 60% of C-Suite executives – ahead of cost savings, scalability, ease of maintenance, and speed.

Cloud costs are higher than expected for 6 in 10 organizations (Source: CloudZero)

As companies invest more in the cloud, only 4 in 10 organizations have their cloud costs around where they expect.

Some 490 out of 1000 respondents said their cloud costs were a little higher than where they should be while 110 reported cloud costs were way too high.



BITS Pilani, Pilani Campus



Cloud – Origins & Motivation



Origins

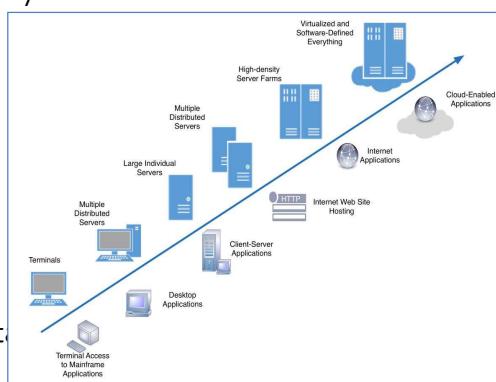
We are at a major inflection point in computing today.

Traditional computational models are now passé.

Amazon started the concept by renting spare computing power from their retail business.

Web and mobile technologies have resulted in information explosion. This means traditional computing power is not enough to process data.

To solve this challenge, we need to use large scale distributed networks. These distributed networks evolved to the Cloud technology.

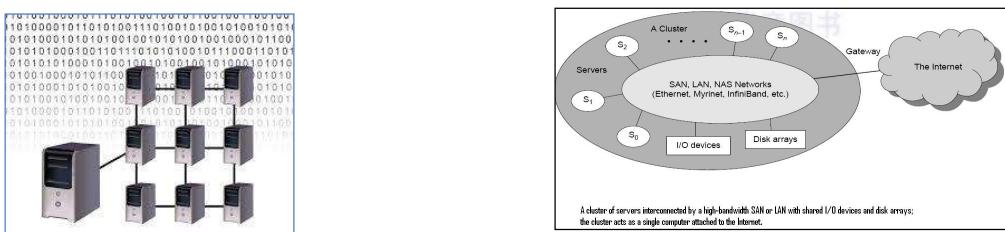


Paradigms

A *paradigm* is a standard, perspective, or set of ideas. A *paradigm* is a way of looking at something.

Types of Distributed Computing

- **Parallel Computing** : Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel")
- **Cluster** : A cluster is a group of loosely coupled computers that work together closely, so that in some respects they can be regarded as a single computer.
- **Grids** : Grid computing is the most distributed form of parallel computing. It makes use of computers communicating over the Internet to work on a given problem.



BITS Pilani, Pilani Campus

Paradigms

A *paradigm* is a standard, perspective, or set of ideas. A *paradigm* is a way of looking at something.

Cloud is the convergence of several traditional computing technologies

Web Technologies

Web Services

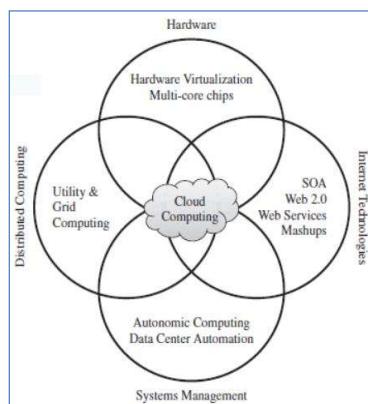
SOA (Service Oriented Architectures)

Distributed Computing

Grids

Clusters

We are now experiencing computing as a service provided by professional organizations, which is served through the medium of high speed internet.

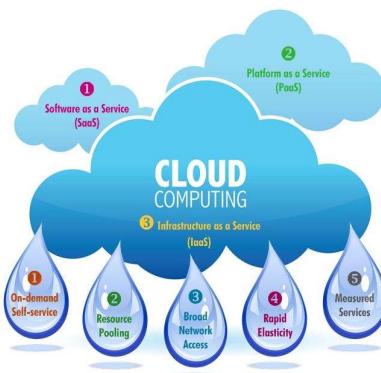


BITS Pilani, Pilani Campus

How do we define Cloud ?

To summarize, cloud computing is the result of the mash-up of several existing disparate technologies, which were progressively modified to suit contemporary computing requirements.

- The applications and services that run on a distributed network using virtualized resources and accessed by common Internet protocols and networking standards comes under Cloud computing. Cloud computing converts the technology, services, and applications that are similar to those on the Internet into a self-service utility. Communicate and coordinate actions by passing messages.
- Cloud computing is based on the concept of pooling physical resources and presenting them as a virtual resource. This computing model supports a new way of provisioning resources, staging applications and for using applications.
- It's bringing computing on an internet scale.



BITS Pilani, Pilani Campus



Terms to Remember

Cloud computing solely exists because of Virtualization technology & Abstraction

Abstraction :

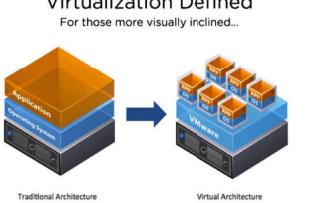
- The details of system implementation is hidden from users and developers.
- Applications run on unspecified physical systems with unknown locations for data, with outsourced system administration of systems.



Virtualization:

- The resources are pooled and shared among the users giving them the illusion that they are the sole owner of the resource. Also resources scales up/down in really short time and without human intervention, charged on metered basis, with multi-tenancy support.

Virtualization Defined



BITS Pilani, Pilani Campus

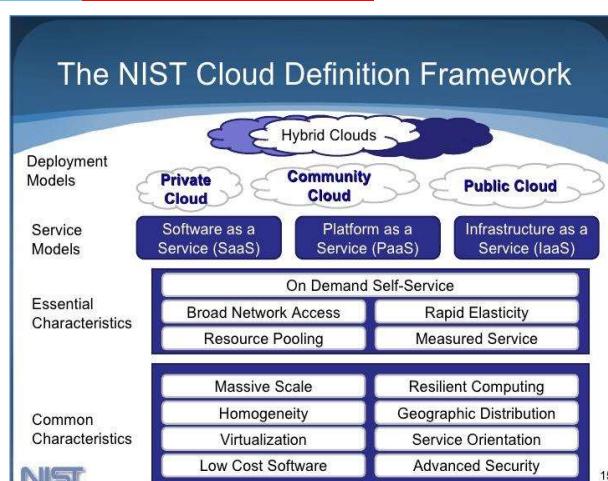


Cloud – NIST & the AAS es



NIST 3-4-5 Rule for Cloud

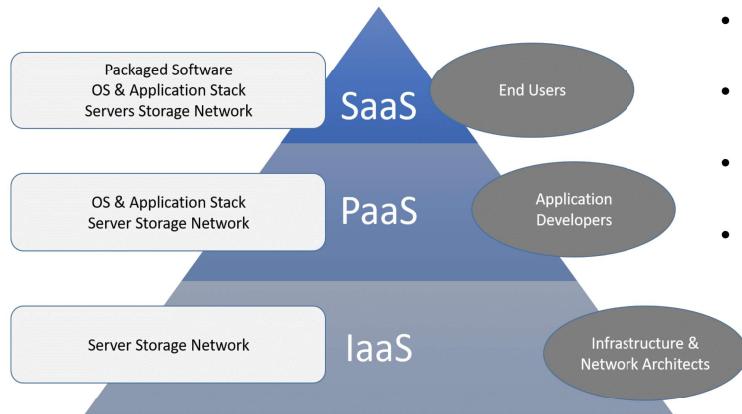
- **3** cloud service models or service types for any cloud platform
- **4** Deployment models
- **5** Essential characteristics of cloud computing infrastructure



The **applications** and **services** that run on a **distributed network** using **virtualized resources** and accessed by **common Internet protocols** and networking standards comes under Cloud computing.

Cloud Service Models (AAS'es)

Cloud Service Models



- There are **3** service models
- **Infrastructure as a Service**
- **Platform as a Service**
- **Software as a Service**

BITS Pilani, Pilani Campus



Infrastructure as a Service

Capability

IaaS provides the following

- Servers- compute, machines
- Storage
- Network
- Operating system

Why IaaS



Characteristics

Resources are distributed as a service

- Allows dynamic scaling (1...10....100....)
- Has a variable costs-
- Generally includes multiple-users on a single piece of hardware. (**multi-tenancy**)

Enabler : Virtualization Technology

- ✓ Manageability and Interoperability
- ✓ Availability and Reliability
- ✓ Scalability and Elasticity

Models

IaaS can be obtained as

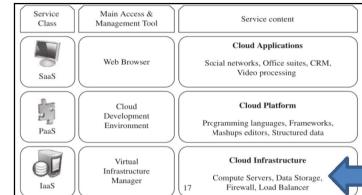
- (1) Public or
- (2) Private infrastructure or
- (3) combination of both

Benefit

The user instead of **purchasing** servers, software, **data center space** or network equipment, **rent** those resources as a fully **outsourced** service on-demand model.

Infrastructure as a Service - Definition

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.
- The consumer can deploy and run software, which can include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).
- Offering virtualized resources (computation, storage, and communication) on demand is known as Infrastructure as a Service (IaaS).
- Infrastructure services are considered to be the bottom layer of cloud computing systems .
- Ex : Amazon EC2 : Elastic Compute Cloud, Eucalyptus, GoGrid, Rackspace Cloud



BITS Pilani

Infrastructure as a Service - Applicability

Where IaaS helps

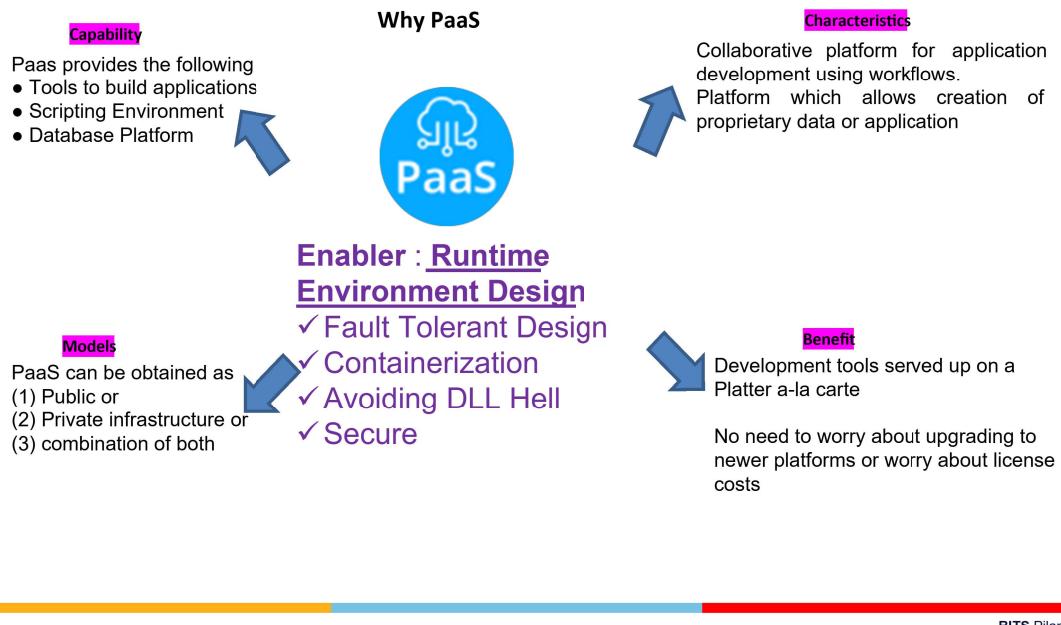
- Where demand is very volatile- encountering spikes and troughs.
- For new enterprise without capital to invest in hardware or entrepreneurs starting on a shoestring budget.
- Where the enterprise is growing rapidly and scaling hardware would be problematic.
- For specific line of business, trial or temporary infrastructural needs
- When you need computing power on the go, turn to IaaS.

What to Do with IaaS

- Test and development.** Teams can quickly set up and dismantle test and development environments, bringing new applications to market faster.
- Website hosting.** Running websites using IaaS can be less expensive than traditional web hosting.
- Storage, backup and recovery.** Organizations avoid the capital outlay. IaaS is useful for handling unpredictable demand and steadily growing storage needs. It can also simplify planning and management of backup and recovery systems
- Big data analysis.** Mining data sets to locate or tease out these hidden patterns requires a huge amount of processing power, which IaaS economically provides.

BITS Pilani

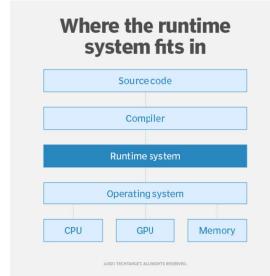
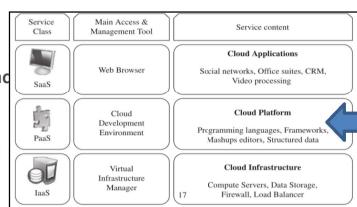
Platform as a Service - Overview



BITS Pilani

Platform as a Service - Definition

- The capability provided to the consumer is to deploy onto the cloud infrastructure, consumer-created or acquired applications created using programming languages and tools supported by the provider.
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.
- A PaaS platform offers an environment on which developers create and deploy applications and do not necessarily need to know how many processors or how much memory that applications will be using.
- In addition, multiple programming models and specialized services (e.g., data access, authentication, and payments) are offered as building blocks to new applications.
- Google AppEngine, Azure, Force.com are examples of Platform as a Service



BITS Pilani

Platform as a Service - Applicability

Where PaaS helps

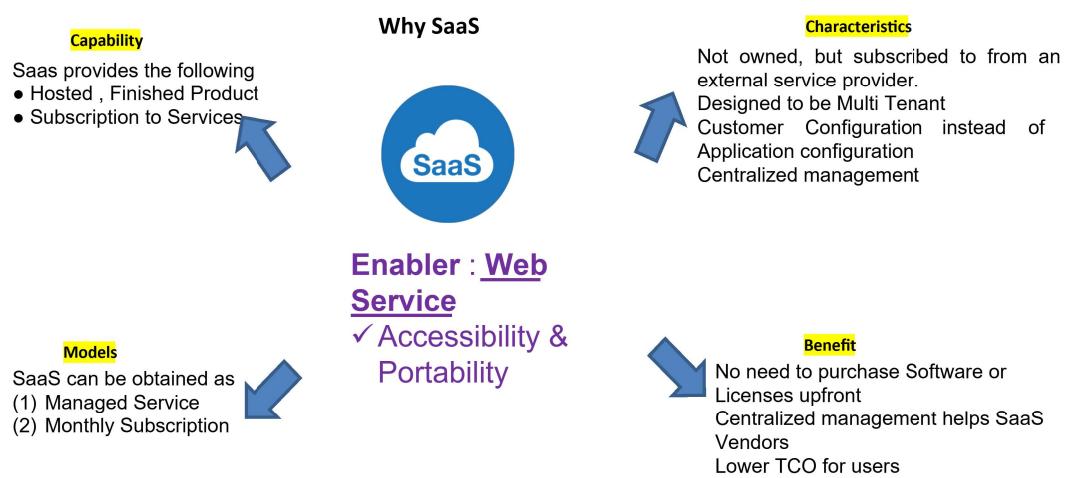
- PaaS allows developers to frequently **change** or **upgrade** operating system features.
- It also helps development teams **collaborate** on projects.
- Security** is provided, including data security and backup and recovery.
- Adaptability**: Features can be **changed** if circumstances dictate 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.

What to Do with PaaS

- Application Services**
 - Services to develop, test, deploy, host and maintain applications in the same integrated development environment.
 - Web based user interface creation tools help to create, modify/test and deploy different UI scenarios
- Multi Tenancy**
 - Construct architecture where multiple concurrent users utilize the same development application
- Collaboration**
 - Support for development team collaboration
 - Tools to handle billing and subscription management

BITS Pilani

Software as a Service - Overview



Software as a Service - Definition

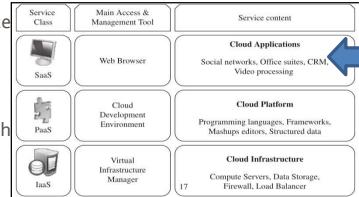
• The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure.

• The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email).

• The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

• This model of delivering applications, known as Software as a Service (SaaS), alleviates the burden of software maintenance for customers and simplifies development and testing for providers.

• [Salesforce.com](#), SaaS model, offers business productivity applications (CRM) that reside completely on their servers, allowing customers to customize and access applications on demand.



BITS Pilani

Which AAS ?

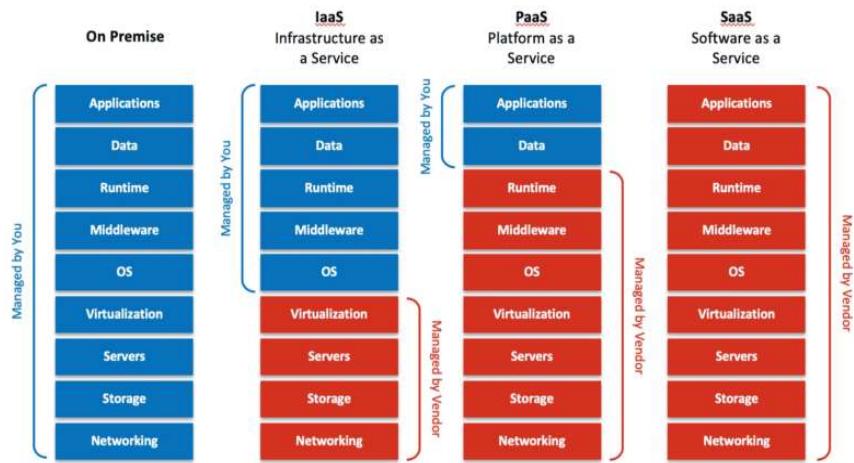
WHICH CLOUD COMPUTING MODEL SHOULD YOU CHOOSE?

IAAS	PAAS	SAAS
Cover infrastructure maintenance and support	Focus on app development instead of infrastructure management	Create solutions with standardized core functionality
Save money and time vs purchasing hardware	Streamline workflows when multiple developers are working on the same project	Develop ecommerce software rapidly, without spending time on server or software issues
Achieve flexibility and scalability when experiencing rapid growth	Rapidly launch an application, reducing costs and time spent on hardware and middleware management	Work on short-term projects and applications that need both web and mobile access

www.apriorit.com

BITS Pilani

Who Manages the AAS es?



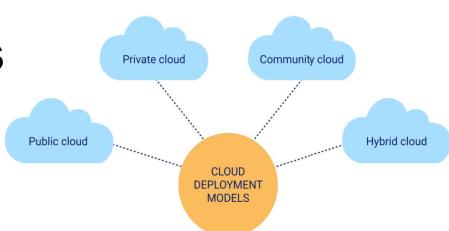
BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Cloud Deployment Models

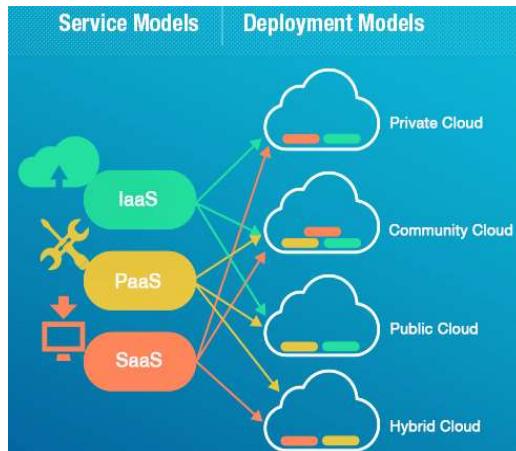


Deployment Models in Cloud

There are four primary cloud deployment models :

- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

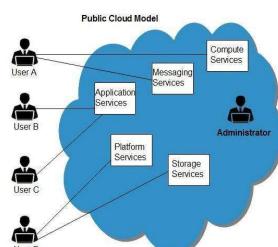
Each can exhibit the previously discussed characteristics; their differences lie primarily in the scope and access of published cloud services, as they are made available to service consumers.



BITS Pilani

Public Cloud

- Public cloud is a cloud infrastructure **owned by a cloud service provider** that provides cloud services to the public for **commercial purposes**.
- Cloud infrastructure available for public consumption on a **pay per use** basis.
- Examples of public clouds include **Amazon Elastic Compute Cloud (EC2)**, IBM's Blue Cloud, Sun Cloud, Google AppEngine and Windows Azure Services Platform.
- **Characteristics**
 - ✓ Homogeneous infrastructure
 - ✓ Common policies , Shared resources and multi-tenant
 - ✓ Leased or rented infrastructure, Economies of scale



BITS Pilani

Public Cloud - Advantage



Cost Effective

- Since **public cloud shares** same resources with large number of customers it turns out **inexpensive**.

Reliability

- The **public cloud** employs large number of **resources** from different locations. If any of the resources fails, public cloud can **employ** another one.

Flexibility

- The public cloud can **smoothly integrate** with **private** cloud, which gives customers a **flexible** approach.

Location Independence

- **Public cloud** services are delivered through Internet, ensuring location independence.

Utility Style Costing

- Public cloud is also based on **pay-per-use** model and resources are accessible whenever customer needs them.

High Scalability

- Cloud resources are made available on demand from a **pool of resources**, i.e., they can be scaled up or down according the requirement.

BITS Pilani

Public Cloud - Disadvantage

Low Security

- In **public cloud model**, data is hosted off-site and resources are shared publicly, therefore does not ensure higher level of security.

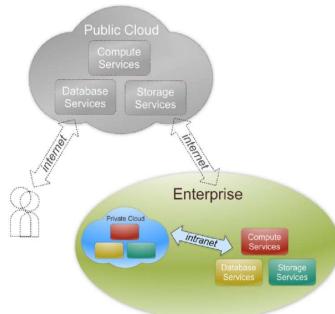
Less Customizable

- It is comparatively less customizable than private cloud.

BITS Pilani

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**.
- Also referred to as **internal cloud** or **on-premise** cloud, a private cloud **intentionally limits access** to its resources to **service consumers** that belong to the same organization that owns the cloud.
- **Characteristics:**
 - Heterogeneous infrastructure
 - Customized and tailored policies
 - Dedicated resources
 - In-house infrastructure
 - End-to-end control



BITS Pilani

Private Cloud - Advantage

High Security and Privacy

- Private cloud operations are not available to general public and resources are shared from distinct pool of resources. Therefore, it ensures high **security** and **privacy**.

More Control

- The **private cloud** has more control on its resources and hardware than public cloud because it is accessed only within an organization.

Cost and Energy Efficiency

- The **private cloud** resources are not as cost effective as resources in public clouds but they offer more efficiency than public cloud resources.



BITS Pilani

Private Cloud - Disadvantage

Restricted Area of Operation

- The private cloud is only accessible locally and is very difficult to deploy globally.

High Priced

- Purchasing new hardware in order to fulfill the demand is a costly transaction.

Limited Scalability

- The private cloud can be scaled only within capacity of internal hosted resources.

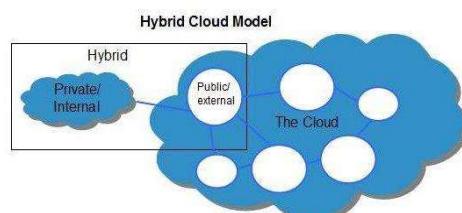
Additional Skills

- In order to maintain cloud deployment, organization requires skilled expertise.

BITS Pilani

Hybrid Cloud

- Hybrid clouds are **mixtures** of these different deployments.
- For example, an enterprise may **rent storage** in a **public** cloud for handling peak demand.
- The **combination** of the enterprise's **private cloud** and the **rented storage** then is a hybrid cloud.
- Clouds retain their **unique identities**, but are **bound together as a unit**.
- A hybrid cloud may offer **standardized** or **proprietary** access to data and applications, as well as application portability.



BITS Pilani

Hybrid Cloud - Advantage

Scalability

- It offers features of both, the public cloud scalability and the private cloud scalability.

Flexibility

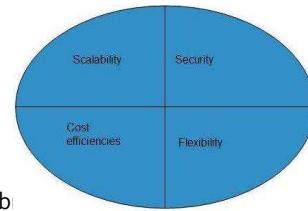
- It offers secure resources and scalable public resources.

Cost Efficiency

- Public clouds are more cost effective than private ones. Therefore, hybrid clouds can be cost saving.

Security

- The private cloud in hybrid cloud ensures higher degree of security.



BITS Pilani

Hybrid Cloud - Disadvantage

Networking Issues

- Networking becomes complex due to presence of private and public cloud.

Security Compliance

- It is necessary to ensure that cloud services are compliant with security policies of the organization.

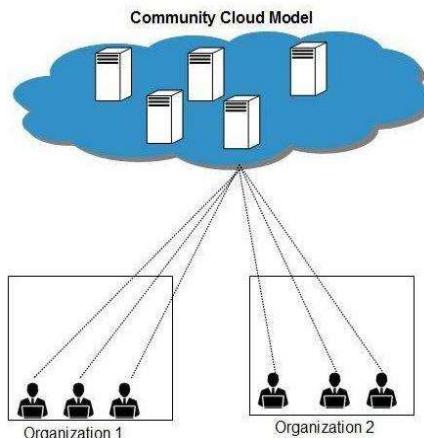
Infrastructure Dependency

- The **hybrid cloud model** is dependent on internal IT infrastructure, therefore it is necessary to ensure redundancy across data centers.

BITS Pilani

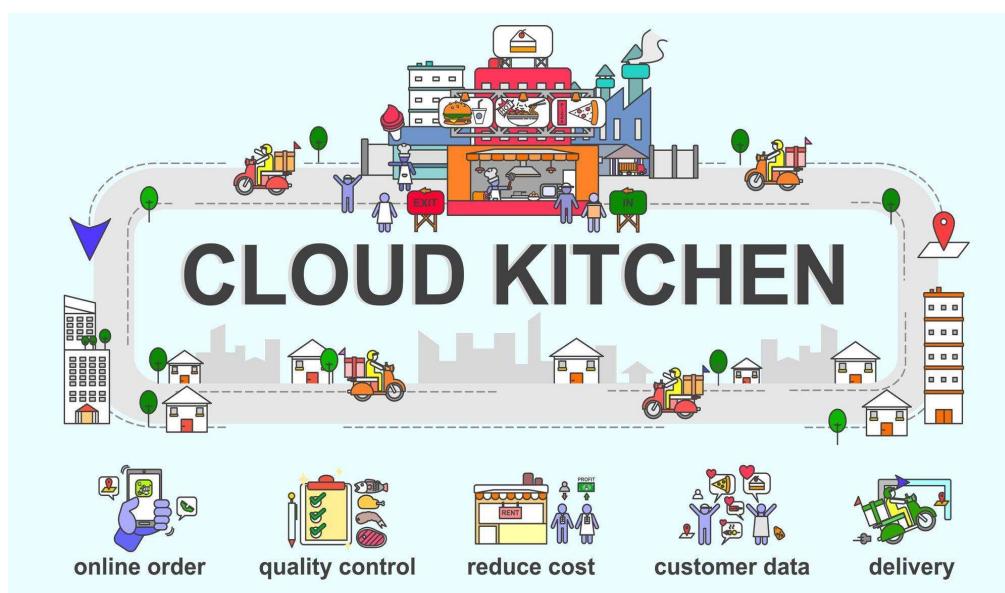
Community Cloud

- Community Cloud is a cloud infrastructure shared by a community of multiple organizations that generally have a common purpose.
- An example of a community cloud is OpenCirrus, which is a cloud computing research testbed intended to be used by universities and research institutions.
- It may be for one organization or for several organizations, but they share common concerns such as their mission, policies, security, regulatory compliance needs, and so on.
- A community cloud may be managed by the constituent organization(s) or by a third party.



BITS Pilani

Community Cloud - Real Life



BITS Pilani

Community Cloud - Advantage

Cost Effective

- Community cloud offers same advantages as that of private cloud at low cost.

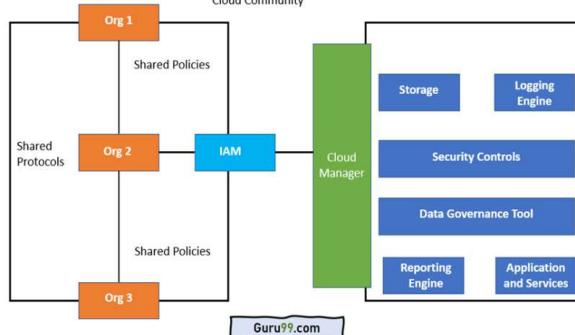
Sharing Among Organizations

- Community cloud provides an infrastructure to share cloud resources and capabilities among several organizations.

Security

- The community cloud is comparatively more secure than the public cloud but less secured than the private cloud.

Community Cloud Architecture



BITS Pilani

Community Cloud - Disadvantage

Logistics Issues

- Since all data is located at one place, one must be careful in storing data in community cloud because it might be accessible to others. It is also challenging to allocate responsibilities of governance, security and cost among organizations.

Control

- Participating organization will not have much control over the infrastructure and configuration options.

Security

- Since it's a community of several organizations, there is high risk that data breaches can occur due to an individual organization's weak security policy.

Integration

- Integration among various systems which are disparate can pose challenges.

BITS Pilani

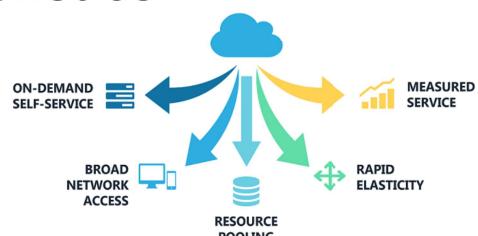
Quick Comparison

Parameters\Type	Public Cloud	Private Cloud	Hybrid Cloud	Community Cloud
Description	In public cloud, services are available for public users.	Private cloud is build up with existing private infrastructure. This type of cloud has some authentic users who can dynamically provision the resources.	Hybrid cloud is a heterogeneous distributed system, resulting from a private cloud, which incorporates different types of services and resources from public clouds.	Different types of cloud are integrated together to meet a common or particular need for some organizations.
Scalability	Very High	Limited	Very High	Limited
Reliability	Moderate	Very High	Medium to High	Very High
Security	Totally Depends on service provider	High class security	Secure	Secure
Performance	Low to medium	Good	Good	Very Good
Cost	Cheaper	High Cost	Costly	Costly
Examples	Amazon EC2, Google AppEngine	VMWare, KVM, Xen	Microsoft, vCloud, Eucalyptus	SolaS Community Cloud, VMWare

BITS Pilani



Cloud Essential Characteristics



Essential Characteristics

5 Essential Characteristics of Cloud Computing

Ref: The NIST Definition of Cloud Computing

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



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

BITS Pilani

Resource pooling

- Cloud services can support **millions of concurrent users**; for example, Skype supports 27 million concurrent users, while Facebook supported 7 million simultaneous users in 2009.
 - Clearly, it is **impossible to support** this number of users if **each user needs dedicated hardware**. Therefore, cloud services need to **share resources between users** and clients in order to **reduce costs**.
- ✓ **Resources** are drawn from a **common pool**.
- ✓ **Common resources** build **economies of scale**.
- ✓ **Common infrastructure** runs at **high efficiency**.
- ✓ Appropriate management of **security & privacy**.

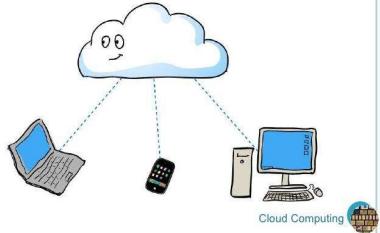


BITS Pilani

Broad Network Access

- **Ubiquitous access** to cloud applications **from desktops, laptops to mobile devices** is critical to the success of a Cloud platform.
 - Thus, **connectivity** is a **critical requirement** for effective use of a **Cloud Application**.
 - For example, cloud services like Amazon, Google, and Yahoo! are available world-wide via the Internet.
 - They are also accessible by a wide variety of devices, such as mobile phones, iPads, and PCs.
- ✓ Users **abstracted** from the implementation
✓ Near **real-time delivery** (seconds or minutes)
✓ Services accessed through a **self-serve web interface**.

Broad Network Access



BITS Pilani

On Demand Self Service

On demand self-service: The compute, storage or platform resources needed by the user of a **cloud platform** are **self-provisioned** or **auto-provisioned** with **minimal configuration**.

For example is possible to log on to Amazon Elastic Compute Cloud (a popular cloud platform) and obtain resources, such as virtual servers or virtual storage, within minutes.

- ✓ Open standards and APIs
- ✓ Almost always IP, HTTP, and REST
- ✓ Available from anywhere with an internet connection



BITS Pilani

Rapid Elasticity

A **cloud platform** should be able to **rapidly increase or decrease computing resources** as needed.

Further, the time taken to provision a new server is very small, on the order of minutes.

- ✓ Resources dynamically-allocated between users.
- ✓ Additional resources dynamically-released when needed.
- ✓ Fully automated

This also increases the speed with which a new infrastructure can be deployed.



BITS Pilani

Metered by Use

One of the **compelling business** use cases for cloud computing is the ability to “**pay as you go**,” where the **consumer pays** only for the **resources** that are **actually used** by his applications.

Commercial cloud services, like Salesforce.com, measure resource usage by customers, and charge proportionally to the resource usage.

- ✓ Services can be cancelled at any time
- ✓ Pay as you go approach



BITS Pilani

Cloud Advantages

Reduced costs : Significant cost reductions are achieved due to higher efficiencies and greater utilization of cloud networks

Ease of utilization: The upfront cost involved in the purchase of hardware and software licenses is lowered a lot. Due to that one can easily make utilization of cloud services.

Quality of Service: Service level agreements with vendor assure the Quality of service

Reliability: The resource scaling and load balancing with fault tolerance capabilities emphasize the high availability of systems.

Outsourced IT management: It results into considerable reduction in IT management complexities and the associated cost.

Simplified maintenance and upgrade: Always latest features are provided to the users removing the need of constant update and up gradations.

Low Entry Barrier: Upfront infrastructure investments are not needed for moving to the cloud.

BITS Pilani

Cloud vs Hosted

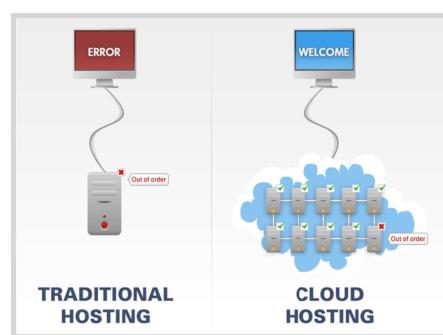
Cloud apps are **web apps** in the sense that they can be used

through **web browsers** but **not all web apps are cloud apps**.

For your **web app** to evolve into a **cloud app**, it should exhibit **certain properties** such as True multi-tenancy to support unique requirements & needs for individual consumers.

Support for virtualization technology, which plays a starring role for cloud era apps.

Web applications should either be built to support this or re-engineered to do so



BITS Pilani

Cloud Challenges

Security and Privacy of Cloud

- ❖ The data store in the cloud must be secure and provide full **confidentiality**. The **cloud provider** should take necessary **security measures** to **secure** the data of the customers.
- ❖ Securities are also the **responsibility of the customer** as they should provide a strong password, should not share the password with anyone, and regularly change the password when we did.
- ❖ **Hacking** can lead to **data loss**; disrupt the encrypted file system and many other problems.



BITS Pilani

Cloud Challenges

Interoperability and Portability

- ❖ The customer must be provided with the services of migration in and out of the cloud – no holds barred.

Reliable and Flexible

- ❖ Reliability and flexibility means that the **data provided** to the cloud should **not leak** and the host **should provide trust to the customers**.
- ❖ To eliminate this challenge the **services provided** by the third party should be **monitored** and **supervision** should be done on **performance, robustness** and business dependency.

Cost

- ❖ Cloud computing is **affordable** but tailor-made deployment based on customer's demand can be **expensive**. Use **Multitenancy to minimize costs**



BITS Pilani

Cloud Challenges

Downtime

- ❖ Downtime is the common challenges of cloud computing as no cloud provider guarantees a platform that is free from downtime. **Apply redundancy and or DR to minimize.**

Lack of resources

- ❖ Lack of resources and expertise is also one of the major challenges faced by the cloud industry and many companies are hoping to overcome this challenge by hiring more workers which are more experienced. **Use Automation**

Management of Multi-Cloud Environment

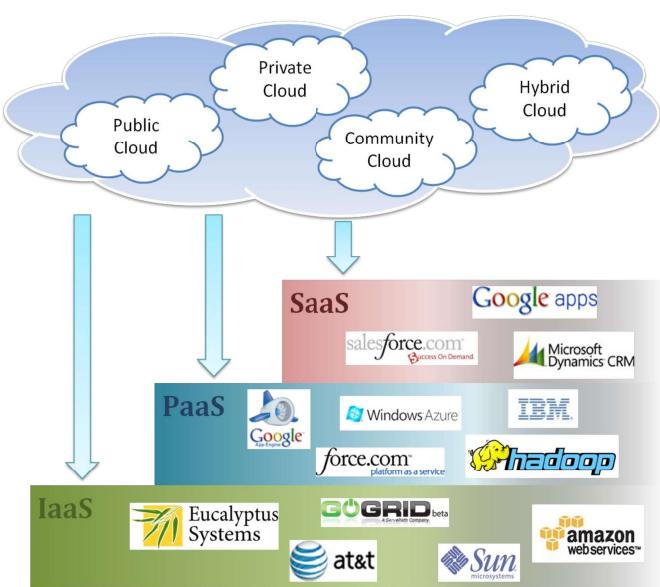
- ❖ Companies nowadays do not use a **single cloud** instead they are using **multiple clouds**. On an average company are using 4.8 different **public** and **private clouds** due to which their management is hindered. **Invest in a good Cloud monitoring tool**



BITS Pilani

Cloud ecosystem

What is cloud computing in your mind Clear or Cloudy?

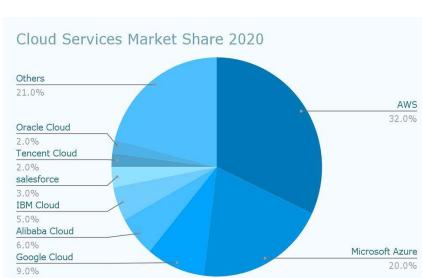
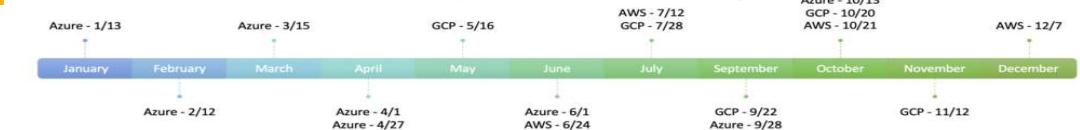


Cloud computing is a new paradigm shift in the way we make use of computing resources. Cloud computing can provide high quality of service at perceived cost benefits. We are moving to an era of computing where we can use code to setup infrastructure on an ad-hoc basis. Service models and deployment models provide services that can be used to

- Rent fundamental computing resources
- Deploy and develop customer-created applications on clouds
- Access provider's applications over network (wired or wireless)

Cloud Failures

2021 Major Public Cloud Outages



Date	Description
12 th Dec	One of the mission-critical AWS cloud units us-east-1 was hit with an outage that took down services like Disney+, Netflix, Slack, Ticketmaster, stock trading app Robinhood, and the crypto exchange Coinbase. Key internal tools like Flex and AtoZ apps used by Amazon warehouse and delivery workers were affected as well.
12 th Nov	Google Cloud went down in mid-November and with it took services like Home Depot, Snap, and Spotify. What caused the outage? A glitch in a network configuration. Yet another scenario showing that betting on a single provider to manage all your apps is pretty risky
20 th Oct	Facebook and its subsidiaries – Messenger, Instagram, WhatsApp, Mapillary, and Oculus – became unavailable for 6 to 7 hours and the world went crazy. Many desperate users flocked to Twitter, Discord, Signal, and Telegram and this resulted in disruptions on these apps' servers.

<https://www.crn.com/slideshows/cloud/the-10-biggest-cloud-outages-of-2021-so-far>

BITS Pilani

Points to Ponder

1. Do you think Cloud is a boon or a bane
2. What is your trust level of any cloud you use
3. What do you Understand by
 1. Hosted Application
 2. Cloud enabled
 3. Cloud native
 4. Cloud Agnostic
4. What do you think will be your main considerations when choosing a cloud provider?

BITS Pilani

Agenda



- ❖ Cloud Recap
 - ❖ What is NIST 3-4-5 Rule
 - ❖ Advantages of Cloud
 - ❖ Disadvantages
- ❖ Introduction to Virtualization
 - ❖ What is Virtualization
 - ❖ Use & demerits of Virtualization
 - ❖ Introducing the Hypervisor
 - ❖ Purpose, Design Goals & Types of Hypervisor
- ❖ Virtualization
 - ❖ Types of Virtualization
 - ❖ X86 Hardware Virtualization
 - ❖ NFV - VNF

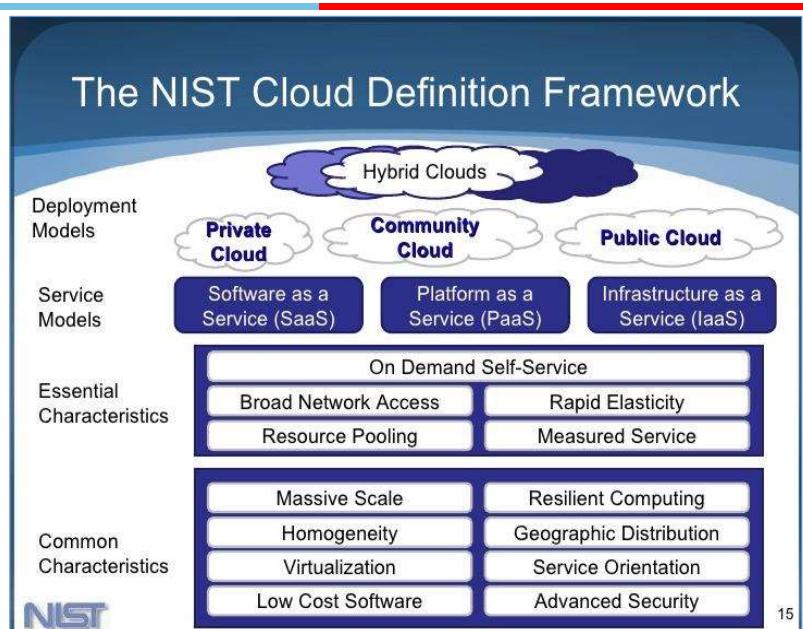
2

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

NIST Definitions

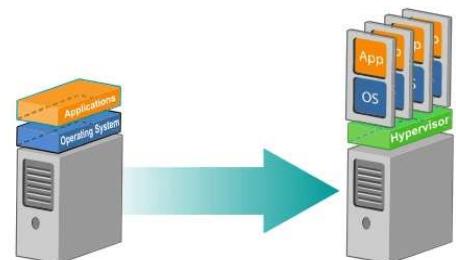


- **3** cloud service models or service types for any cloud platform
- **4** Deployment models
- **5** Essential characteristics of cloud computing infrastructure



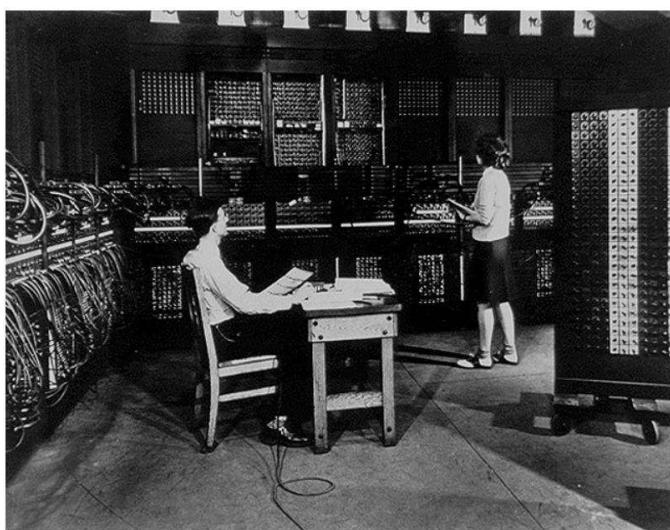


Introduction to Virtualization



Virtualization History

History



ENIAC, the first electronic computing machine **Electronic Numerical Integrator And Computer**



IBM 7094 console, two magnetic tape drives and a punch card reader. This computer took the whole room. © NASA Ames Research Center

Motivations & Origins

Motivation



1 machine → 1 OS → several applications



Applications can affect each other



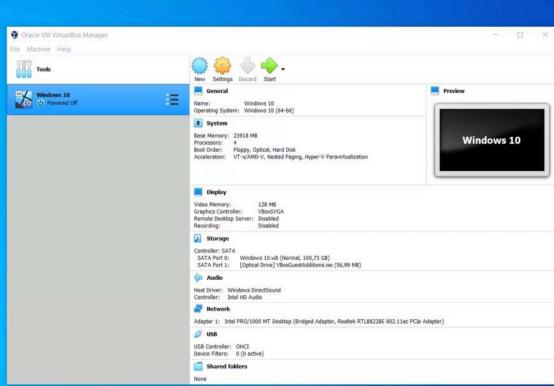
Big disadvantage: machine utilization is very low, most of the times it is below than 25%

Origins

- Server virtualization has existed for several decades
- IBM pioneered more than 30 years ago with the capability to “multitask”
- The inception was in specialized, proprietary, high-end server and mainframe systems. By 1980/90 servers virtualization adoption reduced
- Inexpensive x86 hardware platforms
- Windows/Linux adopted as server

BITS Pilani

Video – Virtualization



Learning Objectives

- Introduce **Oracle Virtual Box**, a hosted hypervisor.
- Demonstrate what a **host system** is what a **guest VM** is and what is the role of the **hypervisor**.
- Students will use the same as **home work** and install virtual box and a choice of their own OS after class.

What is Virtualization?

Virtualization Defined



Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.



Virtualization allows multiple operating system instances to run concurrently on a single computer



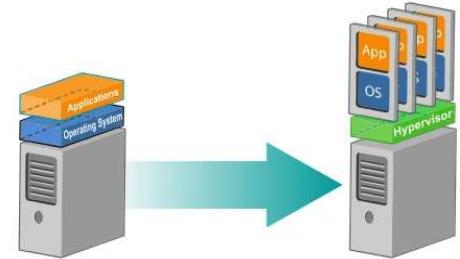
Instead of purchasing and maintaining an entire computer for one application, each application can be given its own operating system, and all those operating systems can reside on a single piece of hardware.



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.

Key Terms:

- VM → Virtual Machine
- VMM → Virtual Machine Monitor
- Hypervisor → VMM
- Multiplexed → Many or several
- Host → System where the VMM resides
- Guest → Virtual Machines created



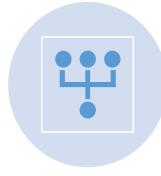
BITS Pilani

What is Virtualization?

Virtualization Objectives



ABSTRACTION – TO SIMPLIFY THE USE OF THE UNDERLYING RESOURCE (E.G., BY REMOVING DETAILS OF THE RESOURCE'S STRUCTURE)



REPLICATION – TO CREATE MULTIPLE INSTANCES OF THE RESOURCE (E.G., TO SIMPLIFY MANAGEMENT OR ALLOCATION)



ISOLATION – TO SEPARATE THE USES WHICH CLIENTS MAKE OF THE UNDERLYING RESOURCES (E.G., TO IMPROVE SECURITY)

Key Terms:

- VM → Virtual Machine
- VMM → Virtual Machine Monitor
- Hypervisor → VMM
- Multiplexed → Many or several
- Host → System where the VMM resides
- Guest → Virtual Machines created

What is Virtualization?

Need of Virtualization

- Cloud can exist without Virtualization, although it will be difficult and inefficient.
- Cloud makes notion of "Pay for what you use", "infinite availability- use as much you want".
- These notions are practical only if we have
 - lot of flexibility
 - efficiency in the back-end.
- This efficiency is readily available in Virtualized Environments and Machines

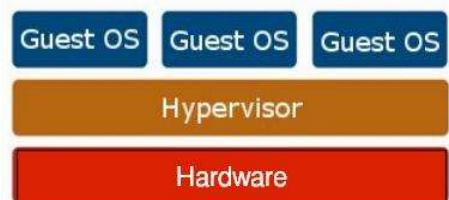
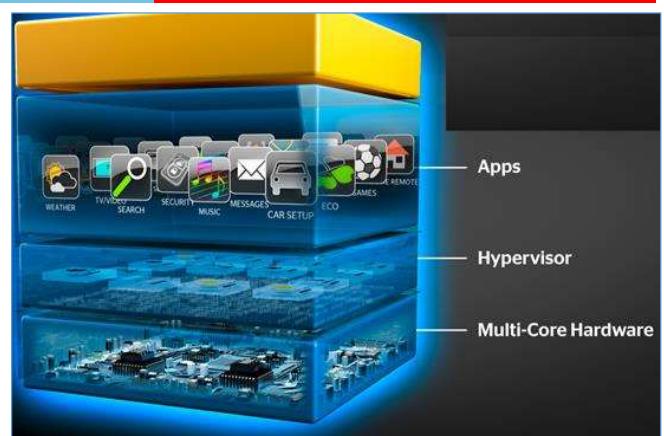
Key Terms:

- VM ➔ Virtual Machine
- VMM ➔ Virtual Machine Monitor
- Hypervisor ➔ VMM
- Multiplexed ➔ Many or several
- Host ➔ System where the VMM resides
- Guest ➔ Virtual Machines created

BITS Pilani

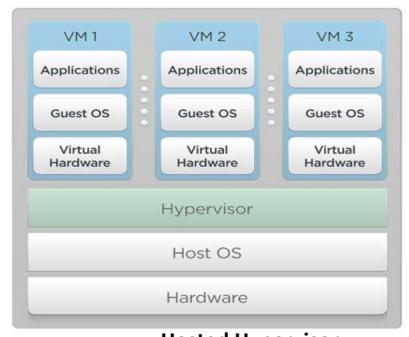
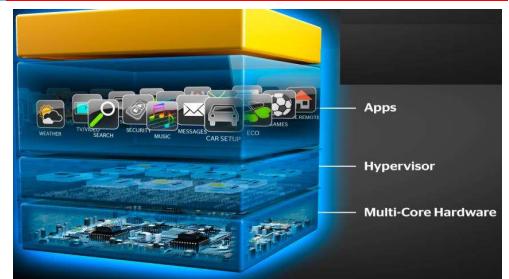
Virtualization Architecture

- 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



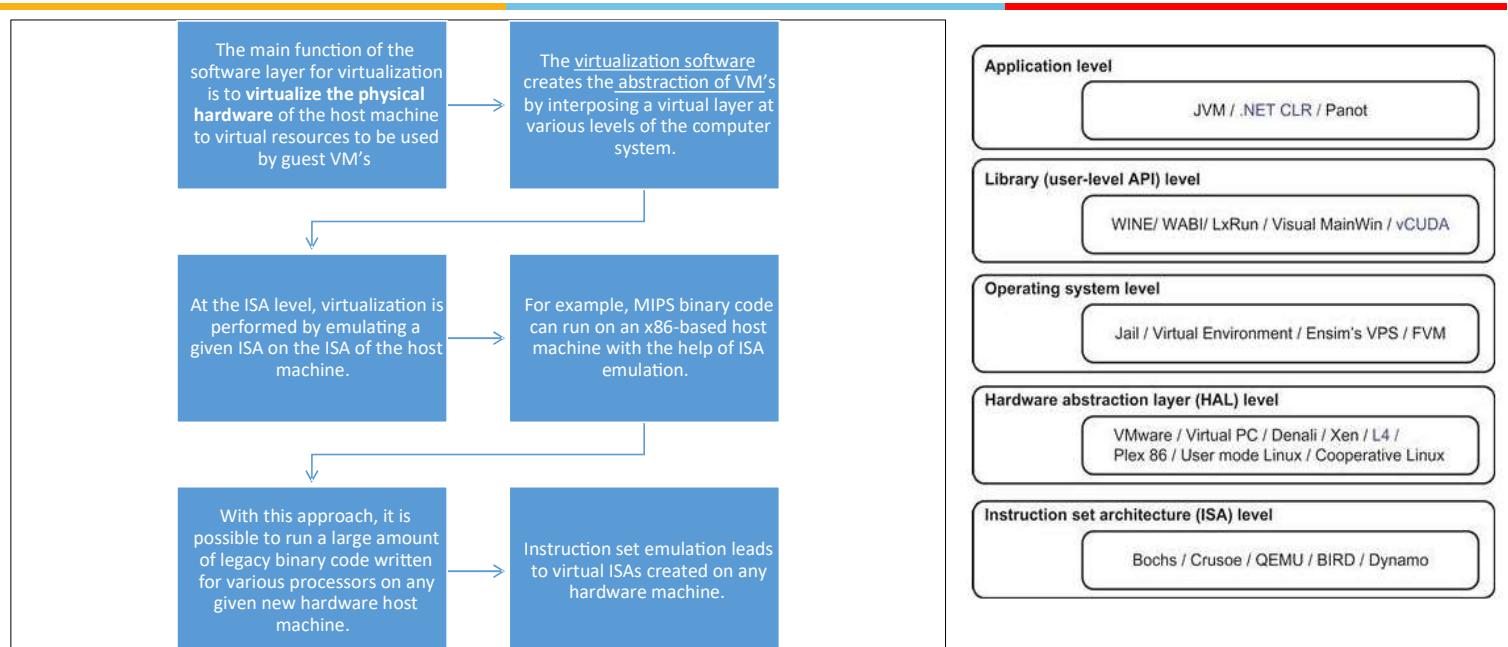
Hypervisor

- A **hypervisor** or **virtual machine monitor (VMM)** is computer software, firmware, or hardware. VMM creates and runs **virtual machines**.
- A computer on which a hypervisor runs one or more virtual machines is called a **host machine**,
- Each virtual machine is called a **guest machine**
- The hypervisor presents the guest systems with **a virtual operating platform** and manages the execution of the guest operating systems.
- Multiple instances of a variety of operating systems may share the virtualized hardware resources:

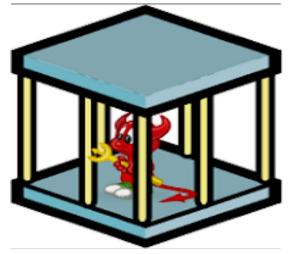


BITS Pilani

Hypervisor Goals



Hypervisor - Samples



- **BOCHS :**

- Bochs is a portable IA-32 and x86-64 IBM PC compatible emulator and debugger mostly written in C++ and distributed as free software under the GNU Lesser General Public License.
- It supports emulation of the processor, memory, disks, display, Ethernet, BIOS and common hardware peripherals of PCs.

- **BSD Jail :**

- The jail mechanism is an implementation of FreeBSD's OS-level virtualisation that allows system administrators to partition a FreeBSD-derived computer system into several independent mini-systems called jails, all sharing the same kernel, with very little overhead.

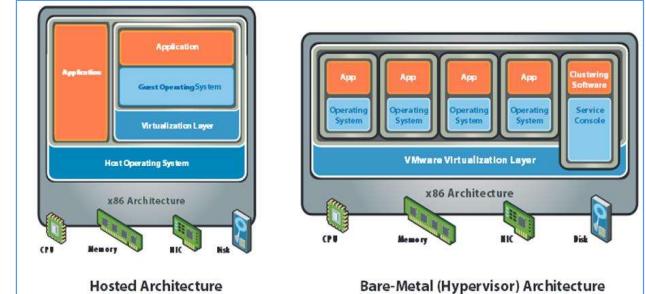
Video – BOCHS

Learning Objectives

- Hypervisors can be used at any abstraction level.
- Oracle Virtual Box was an example of an hardware abstraction.
- We will see Bochs (pronounced Box), which is an ISA abstraction.
- It enables emulation to disparate Instruction sets

Hypervisor Types

- **Hosted:** A hosted architecture installs and runs the virtualization layer as an application on top of an operating system and supports the broadest range of hardware configurations. (VMware Player, ACE)
- **Bare Metal :** The architecture installs the virtualization layer directly on a clean x86-based system. Since it has direct access to the hardware resources rather than going through an operating system, a hypervisor is more efficient than a hosted architecture and delivers greater scalability, robustness and performance. (ESX Server)
- **Hybrid:** The architecture installs the VM layer directly on the hardware like a bare metal, but also leverages the features of the host OS. Xen and Microsoft's Hyper-V are examples of hybrid hypervisors



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

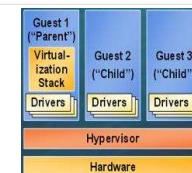
BITS Pilani

Hypervisor Architecture

Monolithic hypervisor

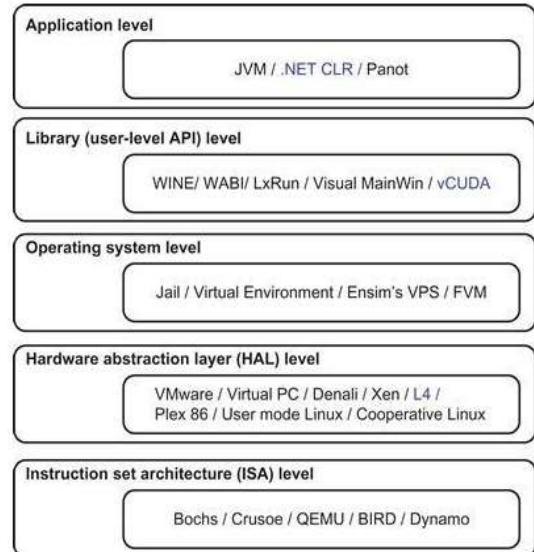
- Simpler than a modern kernel, but still complex
- Contains its own drivers model

BASIS FOR COMPARISON	MICROKERNEL	MONOLITHIC KERNEL
Basic	In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.
Size	Microkernel are smaller in size.	Monolithic kernel is larger than microkernel.
Execution	Slow execution.	Fast execution.
Extendible	The microkernel is easily extendible.	The monolithic kernel is hard to extend.
Security	If a service crashes, it does effect on working of microkernel.	If a service crashes, the whole system crashes in monolithic kernel.
Code	To write a microkernel, more code is required.	To write a monolithic kernel, less code is required.
Example	QNX, Symbian, L4Linux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotes.	Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Microsoft Windows (95,98,Me), Solaris, OS-9, AIX, HP-UX, DOS, OpenVMS, XTS-400 etc.



Comparison

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX



The number of X's in the table cells reflects the advantage points of each implementation level. Five X's implies the best case and one X implies the worst case.

Overall, hardware and OS support will yield the highest performance. However, the hardware and application levels are also the most expensive to implement. User isolation is the most difficult to achieve. ISA implementation offers the best application flexibility.

BITS Pilani

Comparison

Aspect	Type 1 Hypervisor (Bare Metal)	Type 2 Hypervisor (Hosted)
Deployment	Installed directly on the physical hardware	Installed on top of a host operating system
Examples	VMware vSphere/ESXi, Microsoft Hyper-V, Xen	Oracle VirtualBox, VMware Workstation, Parallels Desktop
Performance	Generally higher performance due to direct access to hardware resources	Lower performance due to reliance on host OS for resource allocation
Resource Overhead	Minimal overhead as it operates directly on hardware	Higher overhead due to running within a host OS
Isolation	Better isolation between virtual machines (VMs)	Weaker isolation, as issues in host OS can affect VMs
Security	Generally more secure due to reduced attack surface	Less secure due to reliance on host OS security
Scalability	Typically more scalable for large deployments	Limited scalability compared to Type 1 hypervisors
Management	May require additional management tools	Often easier to manage with GUI interfaces
Disadvantages	<ul style="list-style-type: none"> • Requires dedicated hardware • Potentially higher upfront costs • More complex setup and management 	<ul style="list-style-type: none"> • Performance overhead • Limited scalability • Reduced security compared to Type 1 hypervisors

Resource Sharing in VM - CPU



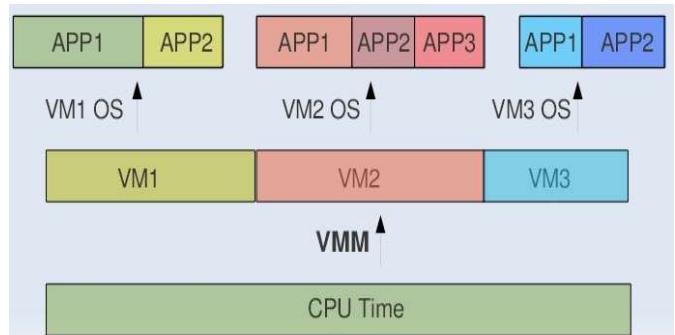
VMM or Hypervisor provides a virtual view of CPU to VMs.



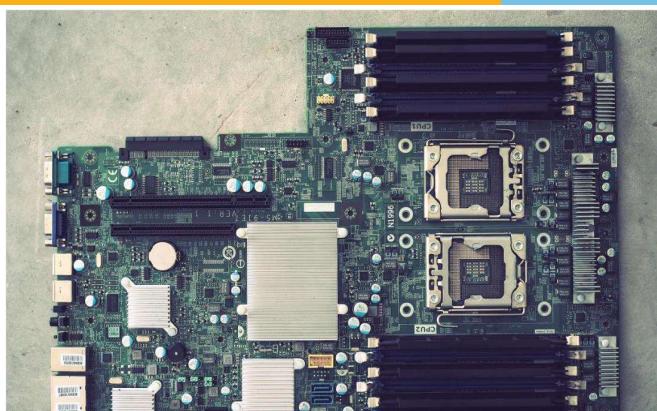
In multi processing, CPU is allotted to the different processes in form of time slices by the OS.



Similarly VMM or Hypervisor allots CPU to different VMs.



Resource Sharing in VM - CPU



A **CPU Socket** is a physical connector on the **motherboard** to which a single physical CPU is connected.

A **CPU** (central processing unit, microprocessor chip, or processor) is a **computer component**. It is the electronic circuitry with transistors that is **connected to a socket**.

A **CPU core** is the **part of a processor(CPU)** containing the L1 cache. The CPU core performs computational tasks **independently without interacting with other cores**, and external components of a "big" processor that are shared among cores. Basically, a core can be considered as a small processor built into the main processor that is connected to a socket. **Applications should support parallel computations to use multicore processors rationally.**

Hyper-threading is a **technology** developed by Intel engineers to bring **parallel computation** to processors that have one processor core. The debut of hyper-threading was in 2002 when the Pentium 4 HT processor was released and positioned for desktop computers. An operating system detects **a single-core processor with hyper-threading** as a processor **with two logical cores (not physical cores)**. Similarly, a four-core processor with hyper-threading appears to an OS as a processor with 8 cores.

A **vCPU** is a virtual processor that is configured as a virtual device in the virtual hardware settings of a VM. A virtual processor can be configured to use multiple CPU cores. A vCPU is connected to a virtual socket.

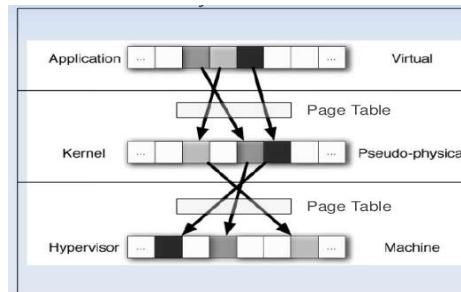
Resource Sharing in VM - Memory



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



Applications use Virtual Addresses

Kernel translates Virtual Addresses to Pseudo-Physical Addresses

Hypervisor translates Pseudo-Physical Addresses to Machine addresses

Memory sharing relies on the observation that several virtual machines might be running instances of the same guest operating system.

These virtual machines might have the same applications or components loaded, or contain common data.

In such cases, a host uses a proprietary Transparent Page Sharing (TPS) technique to eliminate redundant copies of memory pages.

With memory sharing, a workload running on a virtual machine often consumes less memory than it might when running on physical machines.

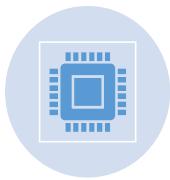
As a result, higher levels of overcommitment can be supported efficiently.

The amount of memory saved by memory sharing depends on whether the workload consists of nearly identical machines which might free up more memory.

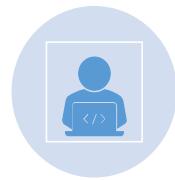
A more diverse workload might result in a lower percentage of memory savings.

BITS Pilani

Resource Sharing in VM - IO



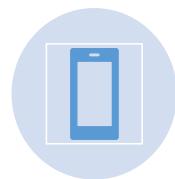
Device needs to use Physical Memory location.



In a virtualized environment, the kernel is running in a hypervisor-provided virtual address space



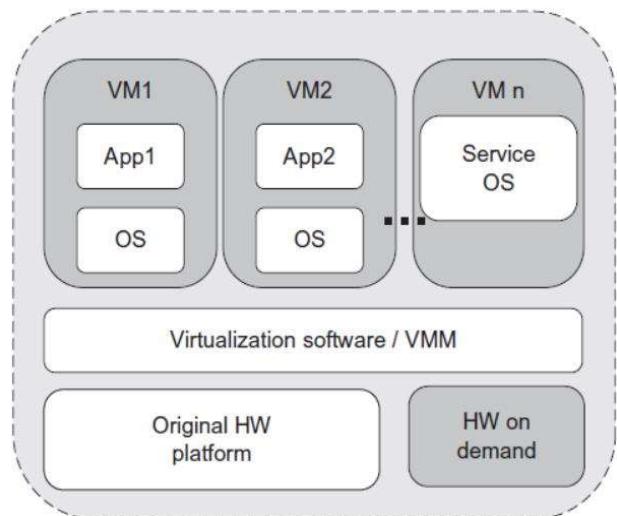
Allowing the guest kernel to convey an arbitrary location to device for writing is a serious security hole



Each device defines its own protocol for talking to drivers

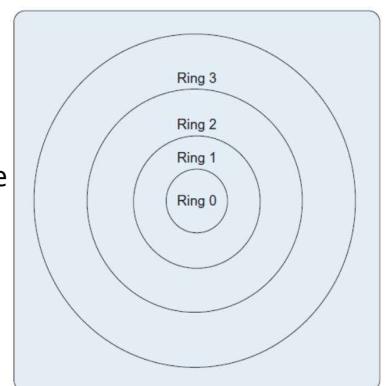
Hypervisor Techniques

- At a very high level, all three types of hypervisors described earlier operate in a similar manner.
- In each case, the guests continue execution until they try to access a shared physical resource of the hardware (such as an I/O device), or an interrupt is received.
- When this happens, the hypervisor regains control and mediates access to the hardware, or handles the interrupt.



Hypervisor Techniques

- To accomplish this functionality, hypervisors rely on a feature of modern processors known as the privilege level or protection ring.
- The basic idea behind privilege levels is that all instructions that modify the physical hardware configuration are permitted at the highest level,
- At lower levels, only restricted sets of instructions can be executed.
- There are four rings, numbered from 0 to 3.
- Programs executing in Ring 0 have the highest privileges, and are allowed to execute any instructions or access any physical resources such as memory pages or I/O devices.
- Guests are typically made to execute in ring 3. This is accomplished by setting Current Privilege Level (CPL) register of the processor to 3 before starting execution of the guest.



Hypervisor Techniques

- If the guest tries to access a protected resource, such as an I/O device, an interrupt takes place, and the hypervisor regains control.
- The hypervisor then emulates the I/O operation for the guest.
- The exact details depend upon the particular hypervisor (e.g., Xen or Hyper-V).
- Note that in order to emulate the I/O operation, it is necessary for the hypervisor to have maintained the state of the guest and its virtual resources



BITS Pilani

Benefits of 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
- 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



28



Virtualization Summary

- 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.

29

BITS Pilani

Key Terms to Remember

Key Terms:

VM : Virtual Machine

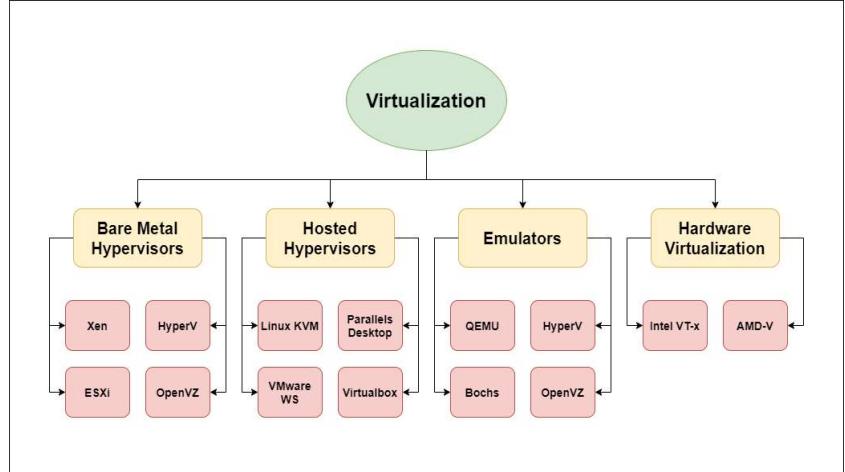
VMM: Virtual Machine Monitor

Hypervisor : VMM

Multiplexed: Many or several

Host: System where the VMM resides

Guest : Virtual Machines created



30

Agenda



- ❖ Virtualization Recap
- ❖ Virtualization Approaches
 - ❖ Motivations
 - ❖ Full Virtualization
 - ❖ Para Virtualization
 - ❖ Hardware Assisted Virtualization
 - ❖ Compare & Contrast architectures
- ❖ X86 Hardware Virtualization
 - ❖ Motivation & Challenges
 - ❖ X86 Hardware Virtualization
 - ❖ NFV - VNF

2

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

What is Virtualization?

Virtualization Defined



Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.



Virtualization allows multiple operating system instances to run concurrently on a single computer



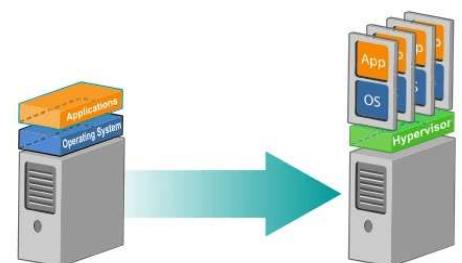
Instead of purchasing and maintaining an entire computer for one application, each application can be given its own operating system, and all those operating systems can reside on a single piece of hardware.



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.

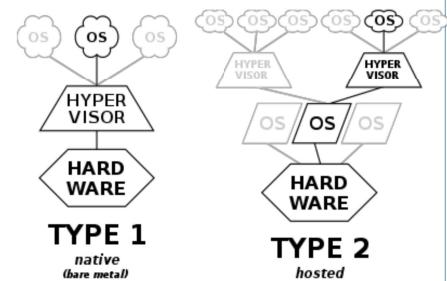
Key Terms:

- VM → Virtual Machine
- VMM → Virtual Machine Monitor
- Hypervisor → VMM
- Multiplexed → Many or several
- Host → System where the VMM resides
- Guest → Virtual Machines created

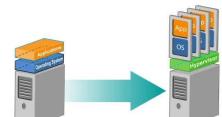




Understanding Hypervisor



What is Hypervisor



Hypervisor Demystified

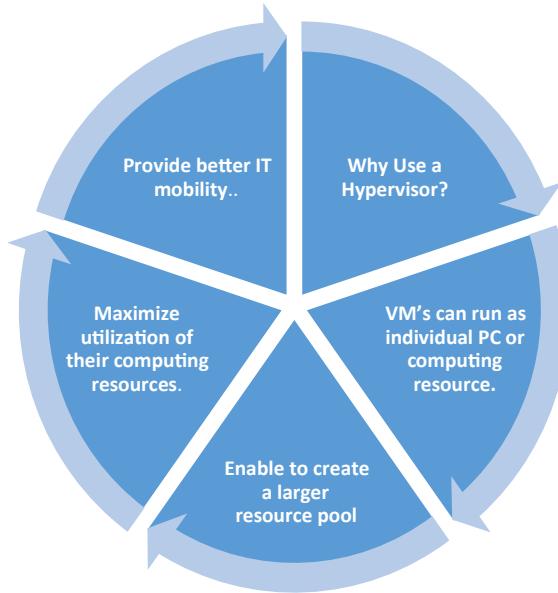
A hypervisor is a form of virtualization software used in Cloud hosting to divide and allocate the resources on various pieces of hardware.

A hypervisor is a crucial piece of software that makes virtualization possible. It creates a virtualization layer that separates the actual hardware components - processors, RAM, and other physical resources - from the virtual machines and the operating systems they run.

Hypervisors emulate available resources so that guest machines can use them. No matter what operating system boots up on a virtual machine, it will think that actual physical hardware is at its disposal..

From a VM's standpoint, there is no difference between the physical and virtualized environment. Guest machines do not know that the hypervisor created them in a virtual environment or that they share available computing power.

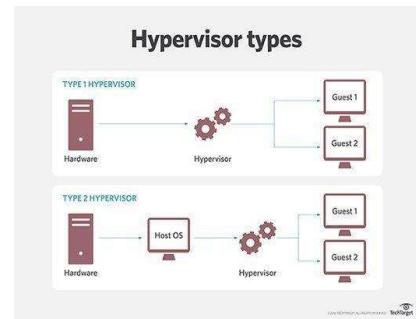
Why use Hypervisor



There are two types of hypervisors, according to their place in the server virtualization structure:

- **Type 1 Hypervisors**, also known as bare-metal or native.
- **Type 2 Hypervisors**, also known as hosted hypervisors.

The sections below explain both types in greater detail.

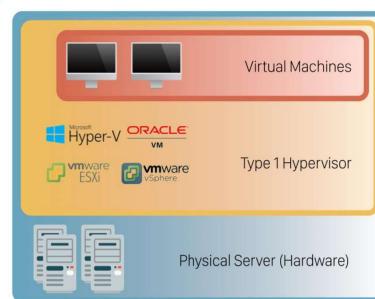


Type 1 Hypervisor

A Type 1 hypervisor is a layer of software installed directly on top of a physical server and its underlying hardware. Since no other software runs between the hardware and the hypervisor, it is also called the [bare-metal hypervisor](#).

This hypervisor type provides excellent performance and stability since it does not run inside Windows or any other operating system. Instead, it is a simple operating system designed to run virtual machines. The physical machine the hypervisor runs on serves virtualization purposes only.

Type 1 hypervisors are mainly found in enterprise environments.



Type 1 Hypervisor – Pros

- **VM Mobility** - Type 1 hypervisors enable moving virtual machines between physical servers, manually or automatically. This move is based on the resource needs of a VM at a given moment and happens without any impact on the end-users. In case of a hardware failure, management software moves virtual machines to a working server as soon as an issue arises. The detection and restoration procedure takes place automatically and seamlessly.
- **Security** - The type 1 hypervisor has direct access to hardware without an additional OS layer. This direct connection significantly decreases the attack surface for potential malicious actors.
- **Resource Over-Allocation** - With type 1 hypervisors, you can assign more resources to your virtual machines than you have. For example, if you have 128GB of RAM on your server and eight virtual machines, you can assign 24GB of RAM to each. This totals 192GB of RAM, but VMs themselves will not consume all 24GB from the physical server. The VMs detect they have 24GB when they only use the amount of RAM they need to perform particular tasks.

Type 1 Hypervisor – Cons

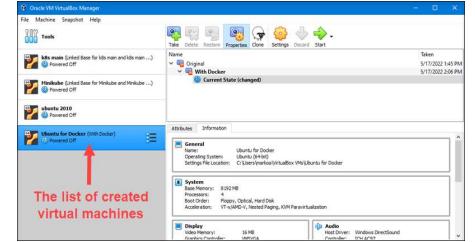
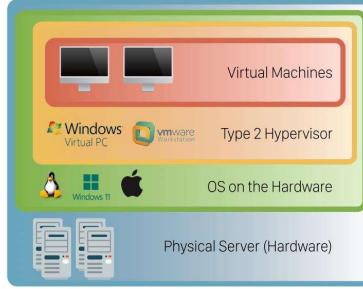
Cons

- **Limited functionality** - Type 1 hypervisors are relatively simple and do not offer many features. The functionalities include basic operations such as changing the date and time, IP address, password, etc.
- **Complicated management** - To create virtual instances, you need a management console set up on another machine. Using the console, you can connect to the hypervisor on the server and manage your virtual environment.
- **Price** - Depending on what functionalities you need, the license cost for management consoles varies substantially.

Type 2 Hypervisor

Type 2 hypervisors run inside the physical host machine's operating system, which is why they are called **hosted hypervisors**. Unlike bare-metal hypervisors that run directly on the hardware, **hosted hypervisors have one software layer in between**. The system with a hosted hypervisor contains:

- A physical machine.
- An operating system installed on the hardware (Windows, Linux, macOS).
- A type 2 hypervisor software within that operating system.
- Guest virtual machine instances.



Type 2 hypervisors are typically found in environments with a small number of servers.

What makes them convenient is that they do not need a management console on another system to set up and manage virtual machines. Everything is performed on the server with the hypervisor installed, and virtual machines launch in a standard OS window.

Hosted hypervisors also act as management consoles for virtual machines. Any task can be performed using the built-in functionalities. Below is one example of a type 2 hypervisor interface (VirtualBox by Oracle):

BITS Pilani

Type 2 Hypervisor - PROs

Pros

- **Easy to manage** - There is no need to install separate software on another machine to create and maintain your virtual environment. Install and run a type 2 hypervisor as any other application within your OS. Create snapshots or clone your virtual machines, import or export appliances, etc.
- **Convenient for testing** - Type 2 hypervisors are convenient for testing new software and research projects. It is possible to use one physical machine to run multiple instances with different operating systems to test how an application behaves in each environment or to create a specific network environment. You only need to ensure that there are enough physical resources to keep the host and virtual machines running.
- **Allows access to additional productivity tools** - The users of type 2 hypervisors can use the tools available on other operating systems alongside their primary OS. For example, Windows users can access Linux applications by creating a Linux virtual machine.

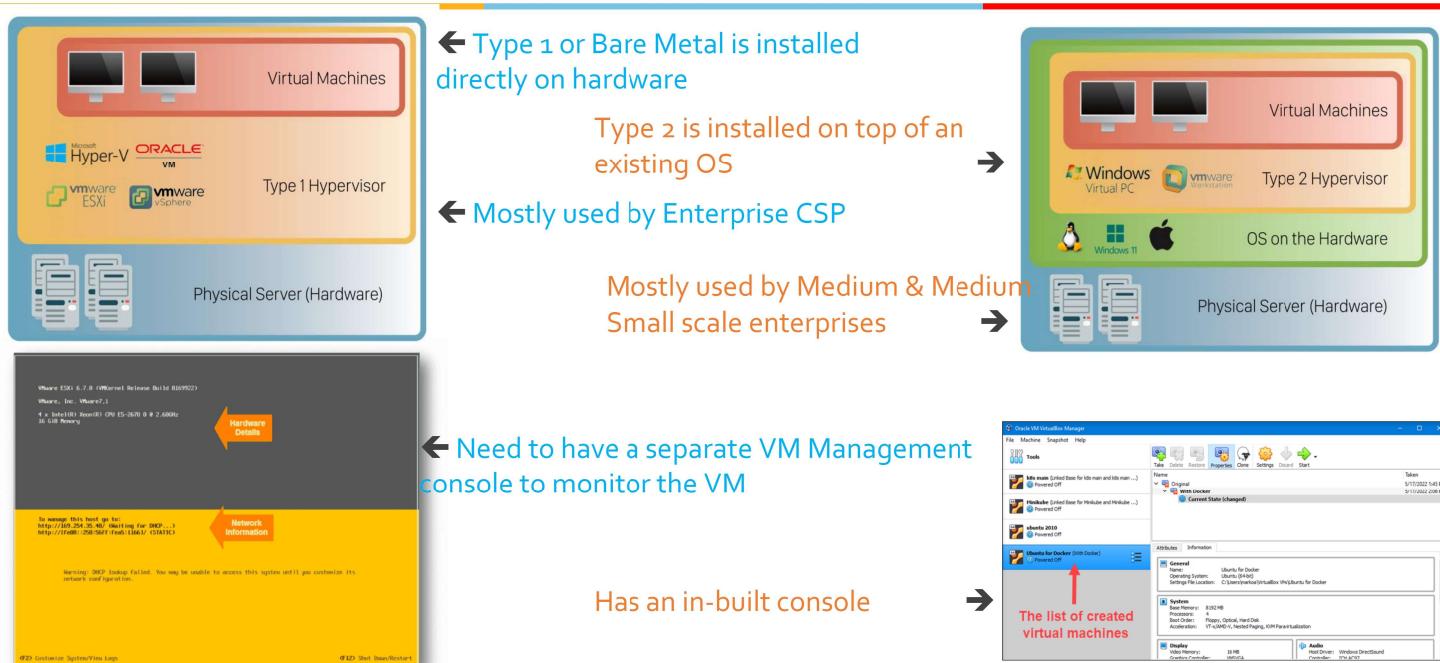
Type 2 Hypervisor - CONs

Cons

- Less flexible resource management** - Allocating resources with this type of hypervisor is more difficult than with type 1. Bare-metal hypervisors can dynamically allocate available resources depending on the current needs of a particular VM. A type 2 hypervisor occupies whatever the user allocates to a virtual machine. When a user assigns 8GB of RAM to a VM, that amount will be taken up even if the VM is using only a fraction of it. If the host machine has 32GB of RAM and the user creates three VMs with 8GB each, they are left with 8GB of RAM to keep the physical machine running. Creating another VM with 8GB of ram would bring down the system.
- Performance** - The host OS creates additional pressure on physical hardware, which may result in VMs having latency issues.
- Security** - Type 2 hypervisors run on top of an operating system. This fact introduces a potential vulnerability since attackers may use potential vulnerabilities of the OS to gain access to virtual machines

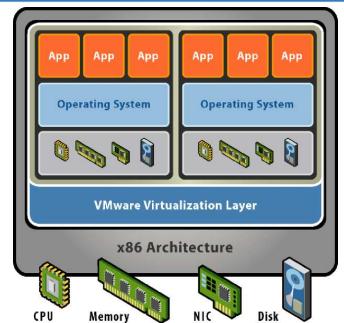
BITS Pilani

Type 1 & 2 Hypervisor – At a Glance

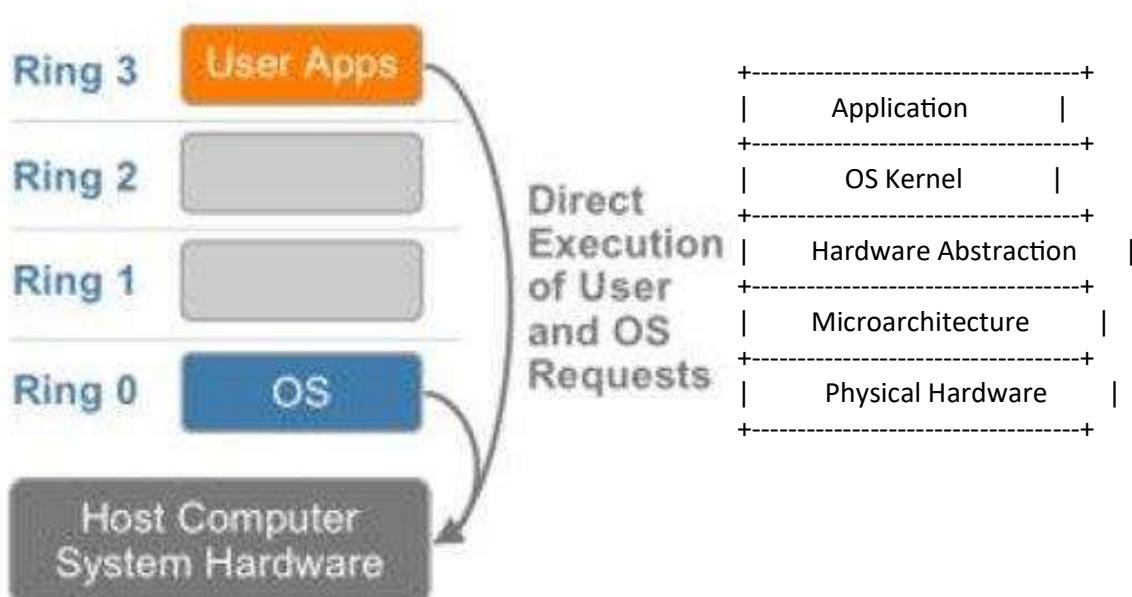




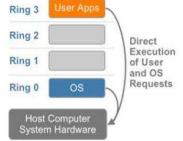
Virtualization Evolution



x86 Architecture



X86 Architecture



The x86 architecture uses a system of **privilege rings** to control access to sensitive areas of the computer's memory and resources. This system is known as **Ring Architecture**, and there are four privilege levels, or rings, numbered 0 through 3.

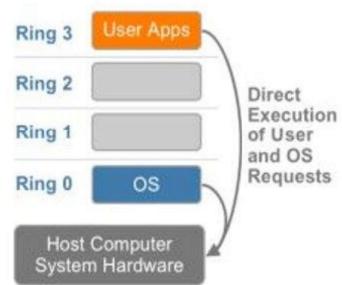
1. **Ring 0:** Also known as the kernel mode, ring 0 is the **most privileged level**, and it has full access to all of the computer's resources, including memory and I/O devices. This is where the operating system kernel runs.
2. **Ring 1:** This ring is used for **device drivers** and other low-level system components. Ring 1 has access to more resources than ring 2 and ring 3, but it still has limited access compared to ring 0.
3. **Ring 2:** Ring 2 is **not used in modern x86 systems**, but it was used in the past for system extensions, such as device drivers and system libraries. Ring 2 has even less access to the computer's resources than ring 1.
4. **Ring 3:** Also known as **user mode**, ring 3 is the least privileged level, and it is where user applications run. Ring 3 has **limited access** to the computer's resources, and any access to sensitive areas of the system must be performed through **system calls** that are handled by the **operating system kernel** running in ring 0.

Each process running on an x86 system runs in a **specific ring**, and the ring level **determines the level of access the process has to the computer's resources**. This system of privilege rings provides a layer of security by ensuring that user applications cannot interfere with the operation of the operating system and other system components.

BITS Pilani

Challenges to Virtualization

- X86 operating systems are designed to run directly on the **bare-metal hardware**, so they naturally assume they fully 'own' the computer hardware.
- As shown, the x86 architecture offers **four levels of privilege** known as **Ring 0, 1, 2 and 3** to operating systems and applications to manage access to the computer hardware.
- While **user level** applications typically run in **Ring 3**, the **operating system** needs to have direct access to the memory and hardware and must execute its privileged instructions in **Ring 0**.
- Virtualizing the x86 architecture requires placing a **virtualization layer** under the operating system (which expects to be in the most privileged Ring 0) to create and manage the **virtual machines** that deliver shared resources



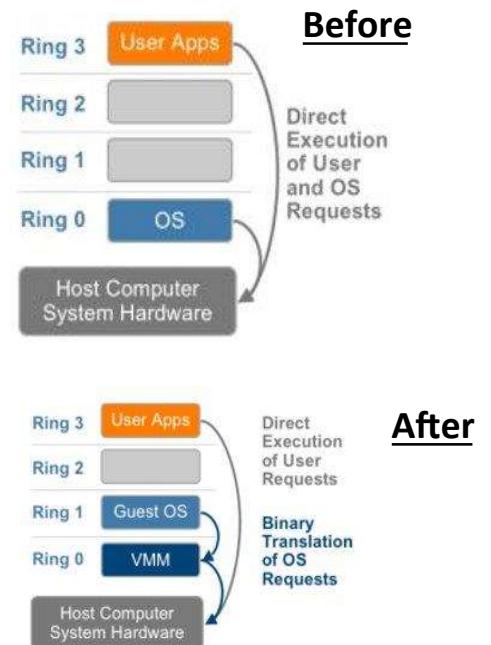
Further complicating the situation, some sensitive instructions can't effectively be virtualized as they have different semantics when they are not executed in Ring 0.

The difficulty in trapping and translating these sensitive and privileged instruction requests at runtime was the challenge that originally made x86 architecture virtualization look impossible



VMware's Solution

- VMware resolved the challenge in 1998, developing binary translation techniques that allow the VMM to run in Ring 0 for isolation and performance,
- The operating system was moved to a user level ring with greater privilege than applications in Ring 3 but less privilege than the virtual machine monitor in Ring 0.
- While VMware's full virtualization approach using binary translation is the de facto standard today the industry as a whole has not yet agreed **on open standards to define and manage virtualization**.
- Each company developing virtualization solutions ~~is free to~~ interpret the technical challenges and develop solutions with varying strengths and weaknesses.



BITS Pilani

Virtualization Evolution

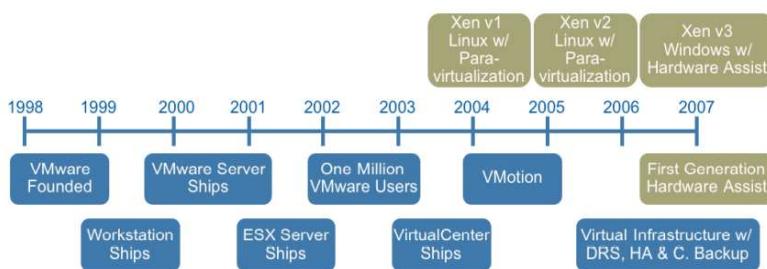


Figure 1 – Summary timeline of x86 virtualization technologies

In 1998, VMware figured out how to virtualize the x86 platform, once thought to be impossible, and created the market for x86 virtualization. The solution was a combination of binary translation and direct execution on the processor

Currently, the **most commonly used virtualization feature is hardware-assisted virtualization**.

Hardware-assisted virtualization is a feature built into modern CPUs that provides improved performance and security for virtualization compared to traditional software-based virtualization technologies.

Hardware-assisted virtualization is supported by Intel's VT-x and AMD's AMD-V technology and is used by many popular virtualization platforms, such as VMware, Hyper-V, and Oracle VirtualBox. This technology allows the **virtualization software** to run **virtual machines** with **near-native performance**, as the **virtualization software** is able to directly access and use the CPU's hardware-assisted virtualization features

that allowed multiple guest OSes to run in full isolation on the same computer with readily affordable virtualization overhead

Introduction

Evolution

1st Generation: Full virtualization (Binary rewriting)

- Software Based
- VMware and Microsoft

2nd Generation: Para virtualization

- Cooperative virtualization
- Modified guest
- VMware, Xen

3rd Generation: Silicon-based (Hardware-assisted) virtualization

- Unmodified guest
- VMware and Xen on virtualization-aware hardware platforms

- **1st Generation: Full virtualization (Binary rewriting)**

- Software Based
- VMware and Microsoft



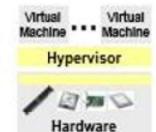
- **2nd Generation: Para virtualization**

- Cooperative virtualization
- Modified guest
- VMware, Xen



- **3rd Generation: Silicon-based (Hardware-assisted) virtualization**

- Unmodified guest
- VMware and Xen on virtualization-aware hardware platforms



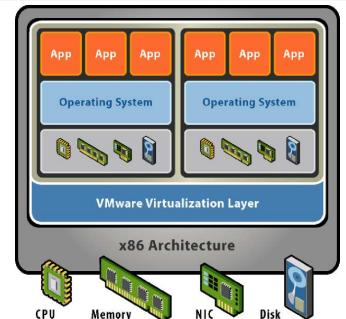
BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Virtualization Approaches



Emulation

Emulation is the process where the virtualizing software mimics that portion of hardware, which is provided to the guest operating system in the virtual machine. The presented emulated hardware is independent of the underlying physical hardware.

Emulation provides VM portability and wide range of hardware compatibility, which means the possibility of executing any virtual machine on any hardware, as the guest operating system interacts only with the emulated hardware.

In an emulated environment, both the application and guest operating system in virtual machines run in the user mode of base operating system. In simple terms, the behavior of the hardware is produced by a software program.

Emulation process involves only those hardware components so that user or virtual machines does not understand the underlying environment.

Emulation

Only CPU & memory are sufficient for basic level of emulation. Typically, emulation is implemented using interpretation. The emulator component takes each and every instruction of user mode and translates to equivalent instruction suitable according to the underlying hardware. This process is also termed as interpretation.

This means that the guest OS remains completely unaware of the virtualization. Also, in interpretation, each and every instruction issued by a VM is trapped in the VMM and interpreted for execution in the hardware. Goes without saying that computationally it is a very expensive method. However, in some cases,, it is needed to use an interpretation technique. However, due to the huge disadvantage of performance, emulation using interpretation is hardly used in virtualization.

Binary Translation / 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.

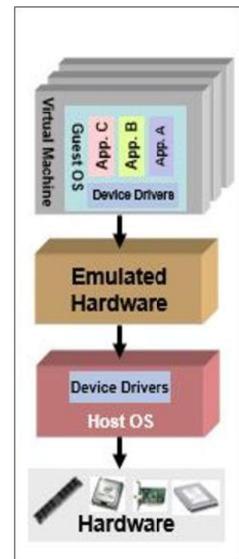
While this is an obvious approach, there are significant performance problems in trying to emulate a complete set of hardware in software.

1st Generation offering of x86/x64 server virtualization

Dynamic binary translation.

The emulation layer talks to an operating system which talks to the computer hardware.

The guest OS doesn't see that it is used in an emulated environment



BITS Pilani

Binary Translation / Full Virtualization - Pros

The emulation layer Isolates VMs from the host OS and from each other.

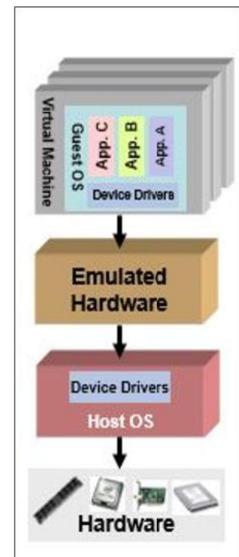
Controls individual VM access to system resources, preventing an unstable VM from impacting system performance.

Total VM portability.

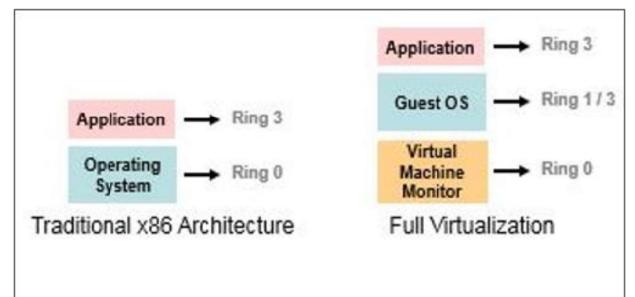
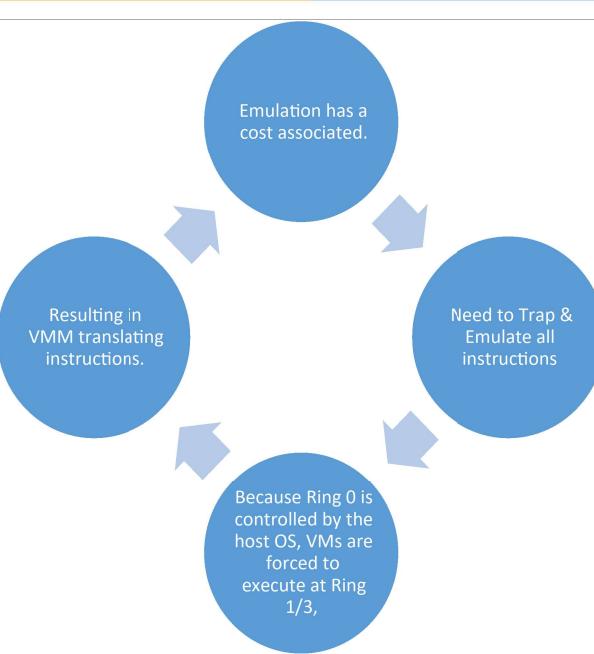
By emulating a consistent set of system hardware, VMs have the ability to transparently move between hosts with dissimilar hardware without any problems.

It is possible to run an operating system that was developed for another architecture on your own architecture.

A VM running on a Dell server can be relocated to a Hewlett-Packard server.



Binary Translation / Full Virtualization - Cons



BITS Pilani

Para Virtualization

"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.

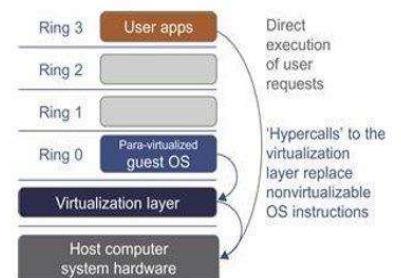
The Guest OS is modified and thus run kernel level operations at Ring 1 (or 3)

the guest is fully aware of how to process privileged instructions

thus, privileged instruction translation by the VMM is no longer necessary

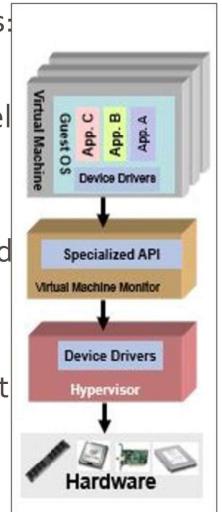
The guest operating system uses a specialized API to talk to the VMM and, in this way, execute the privileged instructions

The VMM is responsible for handling the virtualization requests and putting them to the hardware



Para Virtualization

- Today, VM guest operating systems are para virtualized using two different approaches:
- **Recompiling the OS kernel**
 - Para virtualization drivers and APIs must reside in the guest operating system kernel
 - Modified operating system that includes this specific API, requiring a compiling operating systems to be virtualization aware.
 - Some vendors (such as Novell) have embraced para virtualization and have provided para virtualized OS builds, while other vendors (such as Microsoft) have not.
- **Installing para virtualized drivers**
 - In some operating systems it is not possible to use complete para virtualization, as it requires a specialized version of the operating system
 - To ensure good performance in such environments, para virtualization can be applied for individual devices
 - For example, the instructions generated by network boards or graphical interface cards can be modified before they leave the virtualized machine by using para virtualized drivers



BITS Pilani

H/W Assisted Virtualization

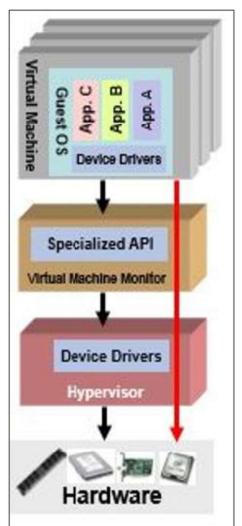
This technique attempts to simplify virtualization because full or paravirtualization is complicated.

Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors.

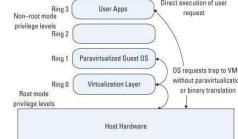
Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring -1.

All the privileged and sensitive instructions are trapped in the hypervisor automatically.

This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating run without modification unlike para virtualization



H/W Assisted Virtualization



Hardware-assisted virtualization. This term refers to a scenario in which the hardware provides [architectural support](#) for building a [virtual machine manager](#) able to run a guest operating system in complete isolation.

This technique was originally introduced in the IBM System/370. At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with *Intel VT* (formerly known as *Vanderpool*) and *AMD V* (formerly known as *Pacifica*).

These extensions, which differ between the two vendors, are meant to reduce the performance penalties experienced by emulating x86 hardware with [hypervisors](#). Before the introduction of hardware-assisted virtualization, software emulation of x86 hardware was significantly costly from the performance point of view.

The reason for this is that by design the x86 architecture did not meet the formal requirements introduced by Popek and Goldberg, and early products were using [binary translation](#) to trap some sensitive instructions and provide an emulated version. Products such as VMware Virtual Platform, introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique. After 2006, Intel and AMD introduced processor extensions, and a wide range of [virtualization solutions](#) took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

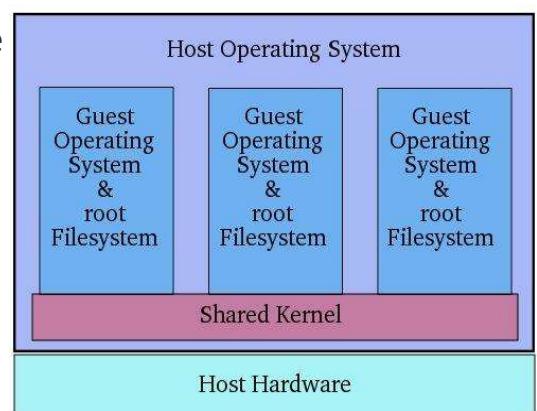
BITS Pilani

SKI Virtualization

Instead of using a hypervisor, it runs a separate version of the Linux kernel and sees the associated virtual machine as a user-space process on the physical host. This makes it easy to run multiple virtual machines on a single host. A device driver is used for communication between the main Linux kernel and the virtual machine.

Processor support is required for virtualization (Intel VT or AMD – v). A slightly modified QEMU process is used as the display and execution containers for the virtual machines. In many ways, kernel-level virtualization is a specialized form of server virtualization.

Examples: User – Mode Linux(UML) and Kernel Virtual Machine(KVM), Docker, LXC



Note:

SKI means Single Kernel Image, is the basis for Container Technologies

Virtualization Comparison

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time.	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	Yes	Yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



Virtualization Types

Virtualization

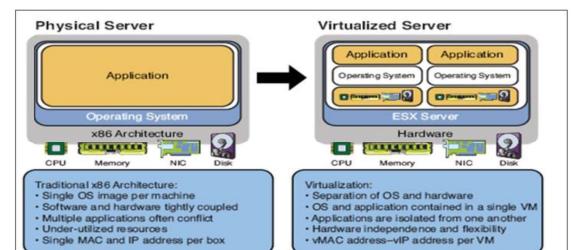
Virtualization

Hardware	Network	Storage	Memory	Software	Data	Desktop
<ul style="list-style-type: none">• Full• Bare-Metal• Hosted• Partial• Para	<ul style="list-style-type: none">• Internal Network Virtualization• External Network Virtualization	<ul style="list-style-type: none">• Block Virtualization• File Virtualization	<ul style="list-style-type: none">• Application Level Integration• OS Level Integration	<ul style="list-style-type: none">• OS Level• Application• Service	<ul style="list-style-type: none">• Database	<ul style="list-style-type: none">• Virtual desktop infrastructure• Hosted Virtual Desktop

BITS Pilani

Server Virtualization

-  Abstracts the physical machine on which the software and operating system is running on and provides an illusion that the software is running on a virtual machine.
-  Enables Infrastructure as a service model.
-  A single physical machine can be used to create several VMs that can run several operating systems independently and simultaneously. VMs are stored as files, so restoring a failed system can be as simple as copying its file onto a new machine.
-  The hypervisor software enables the creation of a virtual machine (VM) that emulates a physical computer by creating a separate OS environment that is logically isolated from the host server.



Server Virtualization - Benefits



Partitioning

Run multiple operating systems on one physical machine.

Divide the physical system resources among virtual machines.

One VM does not know the presence of the other.



Management

Failure of one VM does not affect other VMs.

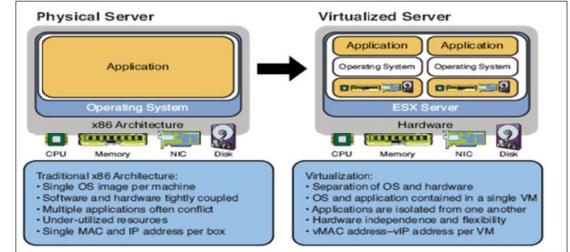
Management agents can be run on each VM separately to determine the individual performance of the VM and the applications that are running on the VM.



Encapsulation

The entire VM state can be saved in a file.

Moving and copying VM information is as easy as copying files.

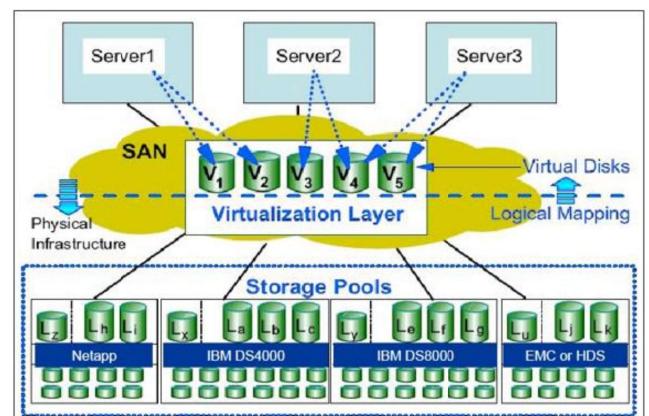


Server virtualization is a key driving force in reducing the number of physical servers and hence the physical space, cooling requirements, cabling, and capital expenses in any data center consolidation projects

BITS Pilani

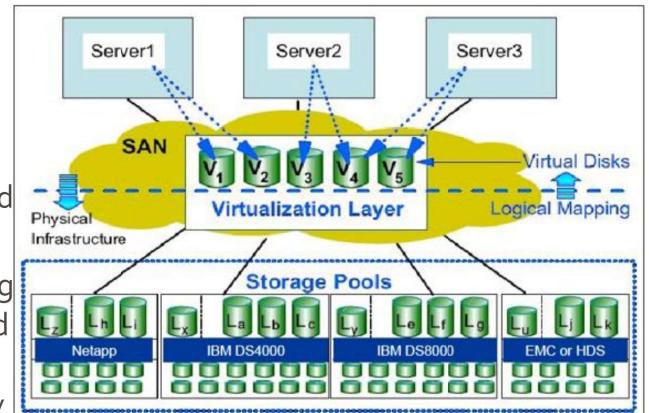
Storage Virtualization

- Storage virtualization refers to providing a logical, abstracted view of physical storage devices.
- It provides a way for many users or applications to access storage without being concerned with where or how that storage is physically located or managed.
- It enables physical storage in an environment to be shared across multiple application servers, and physical devices behind the virtualization layer to be viewed and managed as if they were one large storage pool with no physical boundaries.
- The storage virtualization hides the fact there are separate storage devices in an organization by making all the devices appear as one device.
- Virtualization hides the complex process of where the data needs to be stored and bringing it back and presenting it to the user when it is required.



Storage Virtualization - Benefits

- Typically, the benefits are :-
- **Resource optimization** : Storage virtualization enables you to obtain the storage space on an as-needed basis without any wastage, and it allows organizations to use existing storage assets more efficiently without the need to purchase additional assets.
- **Cost of operation**: Storage virtualization enables adding storage resources without regard to the application, and storage resources can be easily added to the pool by a drag-and-drop method using a management console by the operations people.
- **Increased availability**: Storage virtualization provisions the new storage resources in a minimal amount of time, improving the overall availability of resources
- **Improved performance**



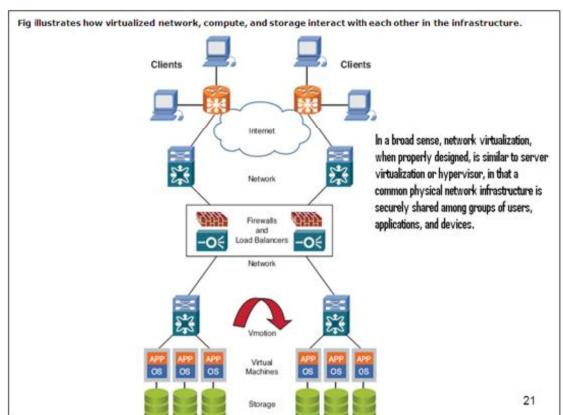
Examples of Storage Virtualization:

- SAN → Storage area Network
- VDA → Virtual Disk Array

BITS Pilani

Network Virtualization

- Network virtualization might be the most ambiguous virtualization of all virtualization types. Several types of network virtualization exist, as briefly described here:
- A **VLAN** is a simple example of network virtualization. VLANs allow logical segmentation of a LAN into several broadcast domains. VLANs are defined on a switch on a port-by-port basis. That is, you might choose to make ports 1–10 part of VLAN 1 and ports 11–20 part of VLAN 2. There's no need for ports in the same VLAN to be contiguous. Because this is a logical segmentation and not physical, workstations connected to the ports do not have to be located together, and users on different floors in a building or different buildings can be connected together to form a LAN.
- Virtual Routing and Forwarding (VRF), commonly used in Multi-Protocol Label Switching (MPLS) networks, **allows multiple instances of a routing table to coexist within the same router at the same time**.



Examples

- VLAN → Virtual Local Area Network
- SDN → S/W Defined Network
- NFV → Network Function Virtualization

Memory Virtualization

Beyond CPU virtualization, the next critical component is memory virtualization.

This involves sharing the physical system memory and dynamically allocating it to virtual machines.

Virtual machine memory virtualization is very similar to the virtual memory support provided by modern operating systems.

Applications see a contiguous address space that is not necessarily tied to the underlying physical memory in the system.

The operating system keeps mappings of virtual page numbers to physical page numbers stored in page tables. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance

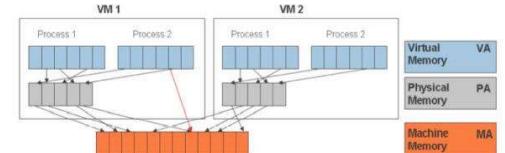


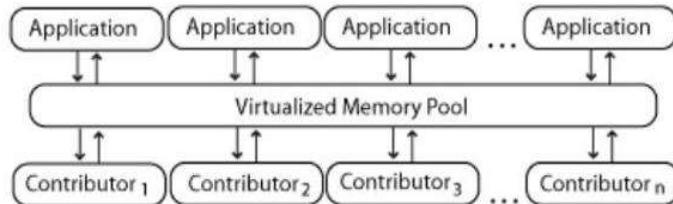
Figure 8 – Memory Virtualization

Memory Virtualization

It introduces a way to decouple memory from the server to provide a shared, distributed or networked function. It enhances performance by providing greater memory capacity without any addition to the main memory. That's why a portion of the disk drive serves as an extension of the main memory.

Application level integration – Applications running on connected computers directly connect to the memory pool through an API or the file system.

Operating System Level Integration – The operating system first connects to the memory pool, and makes that pooled memory available to applications



Device Virtualization

It provides the work convenience and security.

As one can access remotely, you are able to work from any location and on any PC. It provides a lot of flexibility for employees to work from home or on the go.

It also protects confidential data from being lost or stolen by keeping it safe on central servers.

This involves managing the routing of I/O requests between virtual devices and the shared physical hardware.

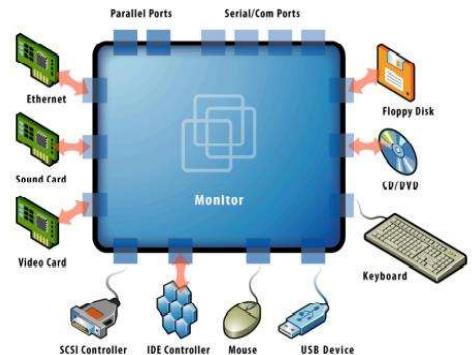


Figure 9 – Device and I/O virtualization

Virtualization Advantages

- 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

Virtualization Summary

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	Yes	Yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

BITS Pilani

Virtualization Advantages

Security: by compartmentalizing environments with different security requirements in different virtual machines one can select the guest operating system and tools that are more appropriate for each environment. For example, we may want to run the Apache web server on top of a Linux guest operating system and a backend MS SQL server on top of a guest Windows XP operating system, all in the same physical platform. A security attack on one virtual machine does not compromise the others because of their isolation.

Virtualization Advantages

Reliability and availability: A software failure in a virtual machine does not affect other virtual machines.

Cost: It is possible to achieve cost reductions by consolidation smaller servers into more powerful servers. Cost reductions stem from hardware cost reductions (economies of scale seen in faster servers), operations cost reductions in terms of personnel, floor space, and software licenses. VMware cites overall cost reductions ranging from 29 to 64%

Virtualization Advantages

Adaptability to Workload Variations: Changes in workload intensity levels can be easily taken care of by shifting resources and priority allocations among virtual machines. Autonomic computing-based resource allocation techniques, such as the ones in can be used to dynamically move processors from one virtual machine to another.

Load Balancing: Since the software state of an entire virtual machine is completely encapsulated by the VMM, it is relatively easy to migrate virtual machines to other platforms in order to improve performance through better load balancing

Virtualization Advantages

Legacy Applications: Even if an organization decides to migrate to a different operating system, it is possible to continue to run legacy applications on the old OS running as a guest OS within a VM. This reduces the migration cost.

Points to Note

- **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.

Application of Virtualization

- Today, virtualization can apply to a range of system layers, including hardware-level virtualization, operating system-level virtualization, and high-level language virtual machines.
- **Maximize resources** — Virtualization can reduce the number of physical systems you need to acquire, and you can get more value out of the servers. Most traditionally built systems are underutilized. Virtualization allows maximum use of the hardware investment.
-
- **Multiple systems** — With virtualization, you can also run multiple types of applications and even run different OS for those applications on the same physical hardware.
- **IT budget integration** — When you use virtualization, management, administration and all the attendant requirements of managing your own infrastructure remain a direct cost of your IT operation.

BITS Pilani

Technology Trends

- Virtualization is Key to Exploiting Trends
- Allows most efficient use of the compute resources
 - Few apps take advantage of 16+ CPUs and huge memory as well as virtualization
 - Virtualization layer worries about NUMA, not apps
- Maximize performance per watt across all servers
 - Run VMs on minimal # of servers, shutting off the others
 - Automated, live migration critical:
 - Provide performance guarantees for dynamic workloads
 - Balance load to minimize number of active servers
- Stateless, Run-anywhere Capabilities
 - Shared network and storage allows flexible mappings
 - Enables additional availability guarantees

52

Agenda



- ❖ Virtualization Recap
- ❖ Infrastructure as a Service
 - ❖ What is IaaS
 - ❖ Introduce AWS
 - ❖ AWS Reference Model
 - ❖ AWS Compute
 - ❖ AWS Storage
 - ❖ AWS Network
 - ❖ AWS Case Study - Abof

2

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

What is Virtualization?

Virtualization Defined



Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.



Virtualization allows multiple operating system instances to run concurrently on a single computer



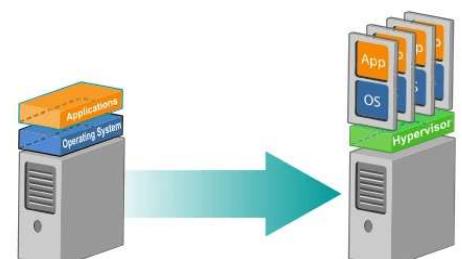
Instead of purchasing and maintaining an entire computer for one application, each application can be given its own operating system, and all those operating systems can reside on a single piece of hardware.



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.

Key Terms:

- VM → Virtual Machine
- VMM → Virtual Machine Monitor
- Hypervisor → VMM
- Multiplexed → Many or several
- Host → System where the VMM resides
- Guest → Virtual Machines created



Virtualization Comparison

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	Yes	Yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

BITS Pilani

Virtualization

Virtualization

Hardware	Network	Storage	Memory	Software	Data	Desktop
<ul style="list-style-type: none"> • Full • Bare-Metal • Hosted • Partial • Para 	<ul style="list-style-type: none"> • Internal Network Virtualization • External Network Virtualization 	<ul style="list-style-type: none"> • Block Virtualization • File Virtualization 	<ul style="list-style-type: none"> • Application Level Integration • OS Level Integration 	<ul style="list-style-type: none"> • OS Level • Application • Service 	<ul style="list-style-type: none"> • Database 	<ul style="list-style-type: none"> • Virtual desktop infrastructure • Hosted Virtual Desktop

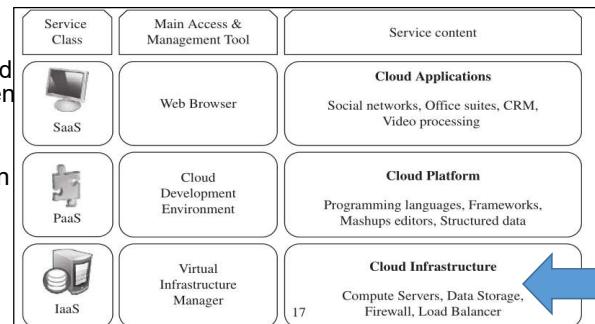


Infrastructure as a Service

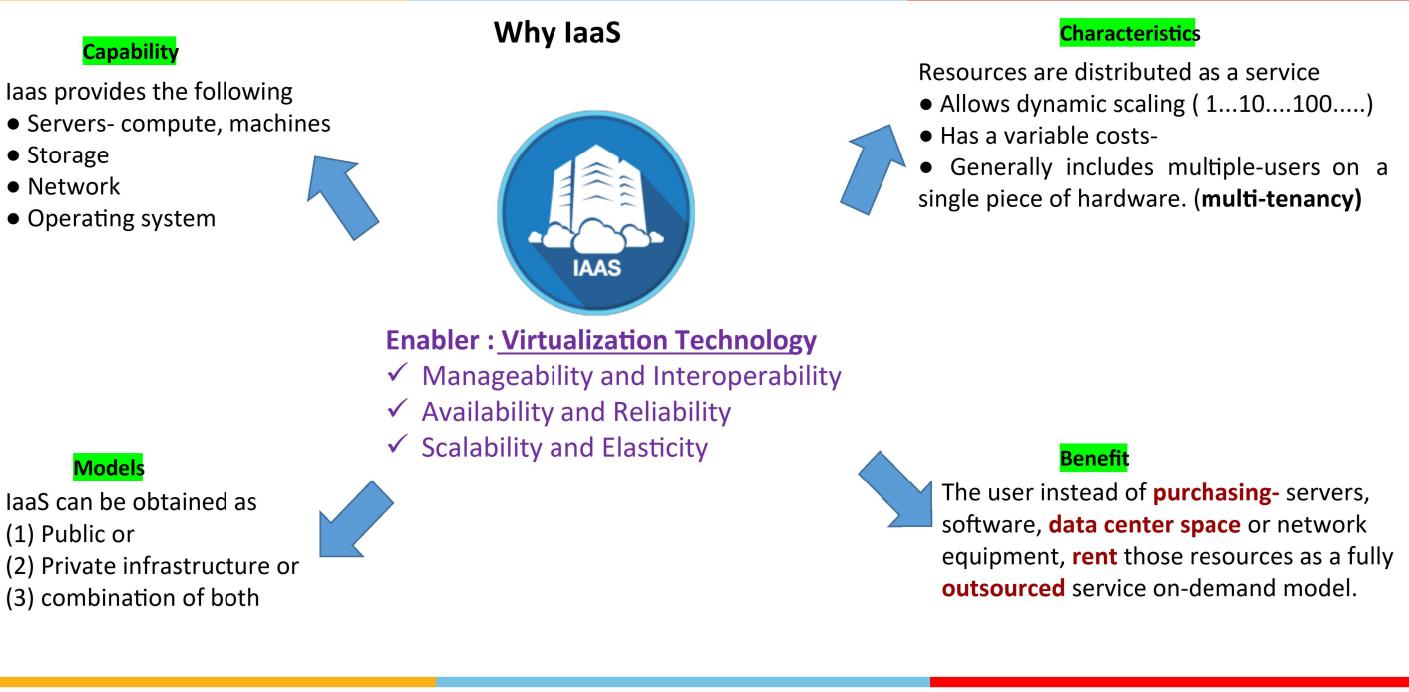


What is IaaS?

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.**
- The consumer is able to deploy and run arbitrary software, which can include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).
- Offering virtualized resources (computation, storage, and communication) on demand is known as Infrastructure as a Service (IaaS).
- Infrastructure services are considered to be the bottom layer of cloud computing systems .
- Ex : **Amazon EC2**: Elastic Compute Cloud, Eucalyptus, GoGrid, Rackspace Cloud

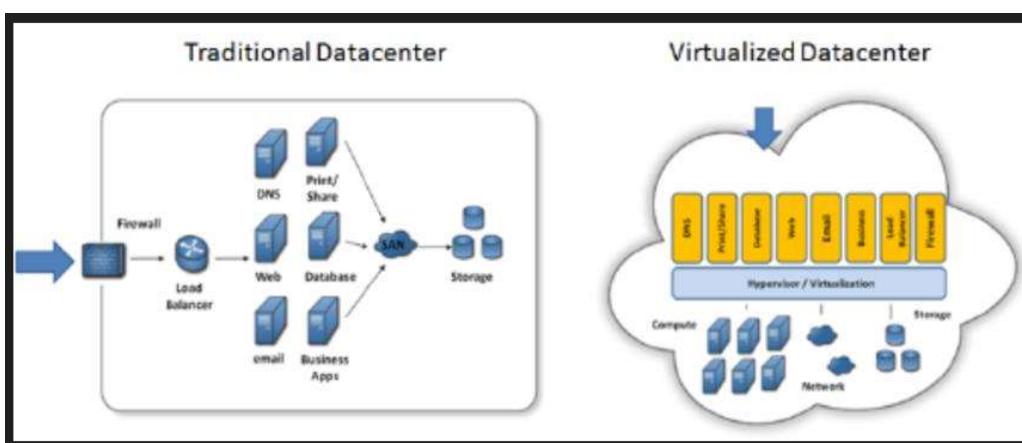


Infrastructure as a Service



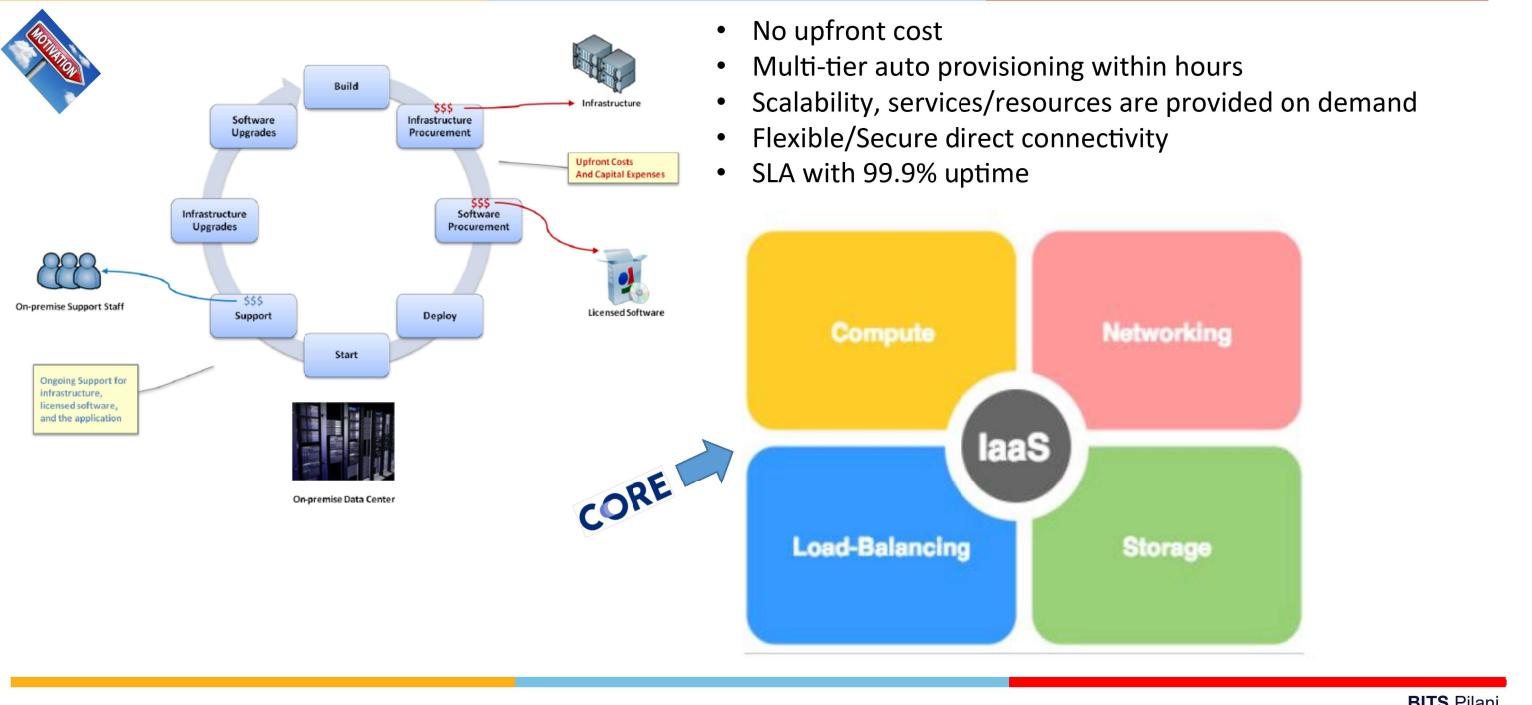
BITS Pilani

Visualization for a New Paradigm



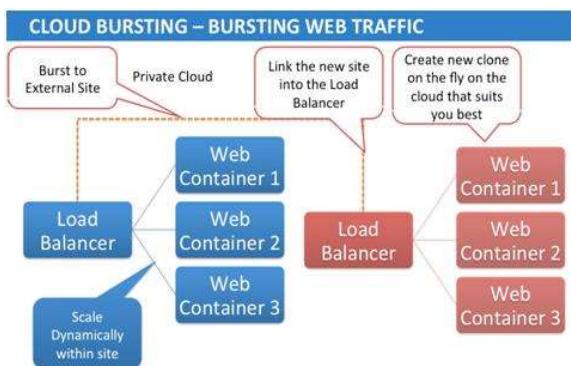
- IAAS , is unlike traditional IT infrastructure.
- End users will not have access to the physical machines.
- Each Hardware resource is provided as a service rather than a physical device
- Flexible/Secure/ Scalable direct connectivity.
- SLA with 99.9% uptime
- Pay as you go policy.
- Payment terms applicable for, resource, bandwidth used, data transferred etc

IaaS - Motivations



BITS Pilani

IaaS – Key Terms



•Telemetry: The process of automatic measurement & transmission of data within the cloud ecosystem to a centralized monitoring unit.

•Cloudbursting: The process of off-loading tasks to the cloud during times when demand exceeds capacity and the most compute resources are needed.

•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 usage** and/or **minimizing risk** to the users.

•Multi-tenant computing: 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](#).

•Hypervisor: Software which enables virtualization.

Pros & Cons of IaaS

IaaS helps

1. Where demand is very **volatile**- encountering **spikes and troughs**.
2. For new enterprise without **capital to invest in hardware or** entrepreneurs starting on a shoestring budget.
3. Where the enterprise is growing rapidly and scaling hardware would be problematic.
4. For specific line of business, trial or temporary infrastructural needs
5. When you need computing power on the go, turn to IaaS.

IaaS Negates

- Where regulatory **compliance** makes the offshoring or outsourcing of data storage and processing difficult
- Where the **highest levels of performance** are required, and on premise or dedicated hosted infrastructure has the capacity to meet the organization's needs



Introducing Amazon Web Service



What is AWS



Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.



Global Infrastructure: AWS serves over one million active customers in more than 190 countries, and it continues to expand its global infrastructure



Security: All AWS customers benefit from data center and network architectures built to satisfy the requirements of the most security-sensitive organizations.

- Application building blocks
- Stable APIs
- Proven Amazon infrastructure
- Focus on innovation and creativity
- Long-term investment

BITS Pilani

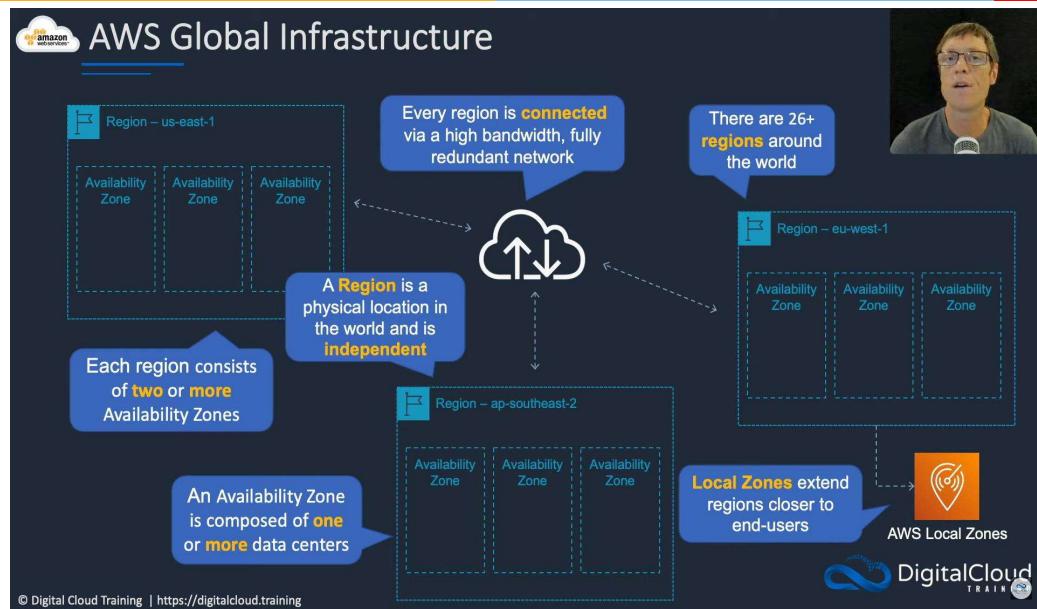
AWS Understanding Service Offering

- AWS operates state-of-the-art, highly available data centers. Although rare, failures can occur that affect the availability of instances that are in the same location.
- If you host all of your instances in a single location that is affected by a failure, none of your instances would be available.
- Amazon EC2 is hosted in multiple locations world-wide. These locations can be further classified as AWS Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones.



Availability Zones located in AWS Regions consist of one or more discrete data centers, each of which has redundant power, networking, and connectivity, and is housed in separate facilities. Each AZ has multiple internet connections and power connections to multiple grids.

AWS Global Infrastructure



AWS Outposts: Creates an AWS infrastructure on the premise of the customer. Provides a seamless hybrid cloud experience

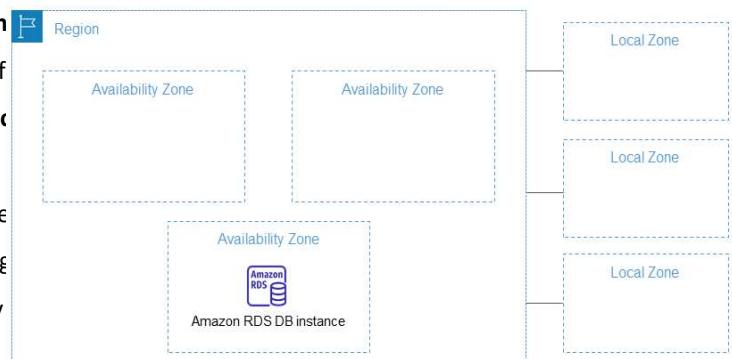
AWS Global Infrastructure

- Services provided via Region, Availability Zone, Local Zone, Wavelength Zone & Outposts
- Region:** Denotes a Physical location across the globe. Currently there are 32 regions.
- Availability Zone:** Physical location within a region where the data centers are built. Currently there are 102+ AZ's
- Local Zone:** Consider this as an extension of an AZ closer to the user's location.
- Wavelength Zone:** Creates an AWS infrastructure on the edge or wireless network of telcos.

BITS Pilani

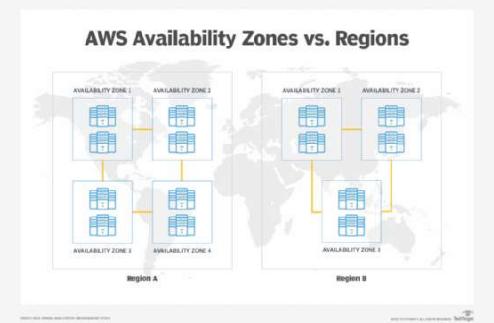
AWS Regions

- AWS provides a **highly available technology infrastructure platform** with multiple **locations worldwide**. These locations are composed of **regions** and **Availability Zones**. Each **region** is a **separate geographic area**.
- It is important to remember that each AWS Region is completely independent. Any Amazon service you initiate (for example, creating database instances or listing available database instances) runs only in your current default AWS Region.
- The default AWS Region can be changed in the console, or by setting the `AWS_DEFAULT_REGION` environment variable. Or it can be overridden by using the `--region` parameter with the AWS Command Line Interface (AWS CLI). **Availability Zones** located in **AWS Regions** consist of one or more **discrete data centers**, each of which has **redundant power**, **networking**, and **connectivity**, and is housed in **separate facilities**.
- Each AWS Region is designed to be isolated from the other AWS Regions. This design achieves the greatest possible fault tolerance to multiple grids and stability.



AWS Availability Zones

- Each Region has multiple, isolated locations known as **Availability Zones**. Each **Availability Zone** is also **isolated**, but the **Availability Zones** in a region are connected through **low-latency links**.
- **Availability Zones** are **physically separated** within a typical metropolitan region and are located in lower-risk flood plains (specific flood zone categorization varies by region). In addition to using a **discrete uninterruptable power supply (UPS)** and **site backup generators**, they are each fed via **different grids** from **independent utilities** (when available) to reduce single points of failure further.
- **Availability Zones** are all **redundantly connected** to multiple tier-1 transit providers. By placing resources in **separate Availability Zones**, you can protect your website or application from a **service disruption** impacting a single location.
- The code for Availability Zone is its Region code followed by a letter identifier. For example, us-east-1a.



- If you **distribute** your instances across **multiple Availability Zones** and one instance fails, you can design your application so that an instance in **another Availability Zone can handle requests**.

BITS Pilani

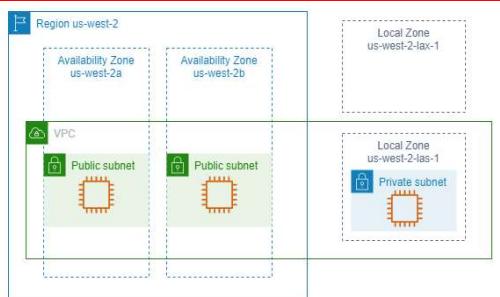
AWS Local Zones

A **Local Zone** is an extension of an **AWS Region** in geographic proximity to your users. **Local Zones have their own connections** to the internet and support AWS Direct Connect, so that resources created in a Local Zone can serve local users with low-latency communications.

The code for a Local Zone is its Region code followed by an identifier that indicates its physical location. For example, us-west-2-lax-1 in Los Angeles.

The **VPC** spans the **Availability Zones** and one of the Local Zones. Each **zone** in the **VPC** has one **subnet**, and each **subnet has an instance**.

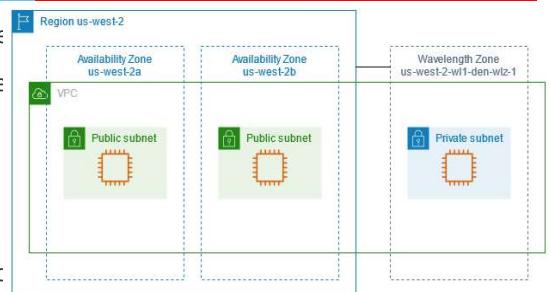
When you **launch an instance**, you can specify **a subnet that is in a Local Zone**. You also allocate an IP address from a network border group. A network border group is a unique set of Availability Zones, Local Zones, or Wavelength Zones from which AWS advertises IP addresses, for example, us-west-2-lax-1a.



- Some **AWS resources** might not be available in all **Regions**. Make sure that you can create the resources that you need in the desired Regions or Local Zones before launching an instance in a specific Local Zone
- Before you can **specify a Local Zone for a resource** or service, you must **opt in** to Local Zones.

AWS Wavelength Zones

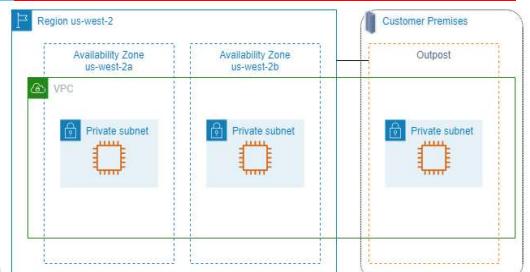
- **Wavelength Zones** are **AWS infrastructure deployments** that embed AWS compute and storage services within **telecommunications providers' data centers** at the edge of the **5G network**, so application traffic can reach application servers running in Wavelength Zones without leaving the mobile providers' network.
- This **prevents the latency** that would result from multiple hops to the internet and enables customers to take **full advantage of 5G networks**. Wavelength Zones extend **AWS to the 5G edge**, delivering a consistent developer experience across **multiple 5G networks around the world**. Wavelength Zones also allow developers to **build** the next generation of **ultra-low latency applications** using the same **familiar AWS services**, APIs, tools, and functionality they already use today.
- Processing at the network edge can help avoid transmitting large volumes of data over the network provider's infrastructure, and offload processing from mobile device hardware.



This enables new classes of compute-intensive, latency sensitive applications latency. For example, a fleet of autonomous cars interacting with road sensors to prevent crashes, smart industrial robots assessing and reacting to plant conditions in a dangerous manufacturing environment, or retailers serving personalized promotions to shoppers' mobile phones in real time as they pass product displays.

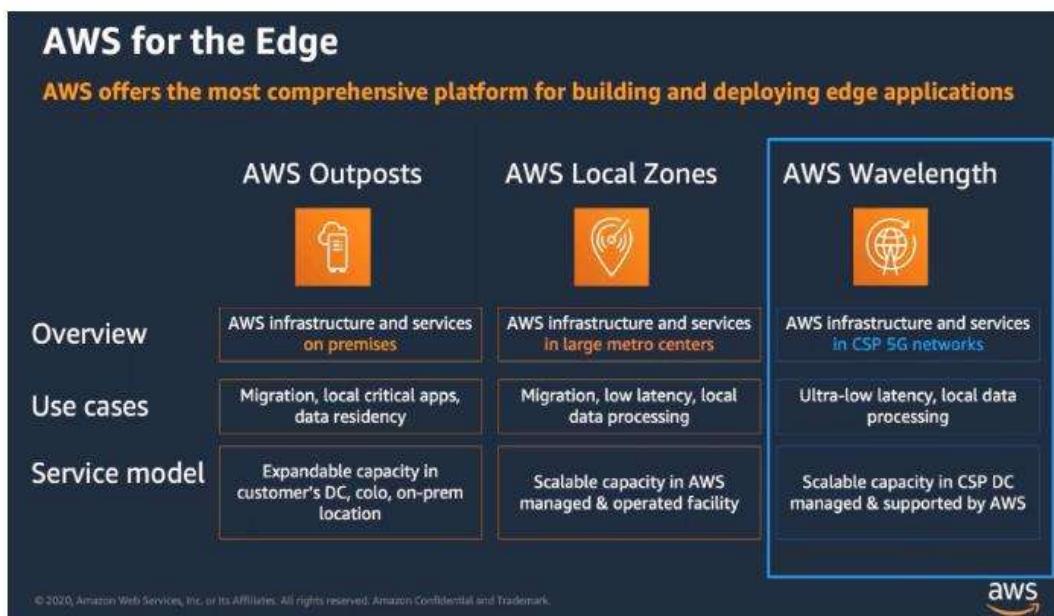
AWS Outposts

- **AWS Outposts** is a fully **managed service** that extends **AWS infrastructure**, services, APIs, and tools to **customer premises**. By providing **local access to AWS managed infrastructure**, AWS Outposts enables customers to build and run **applications** on premises using the same programming interfaces as in AWS Regions, while using local compute and storage resources for lower latency and local data processing needs.
- **AWS operates, monitors, and manages this capacity** as part of an **AWS Region**. You can create subnets on your Outpost and specify them when you create AWS resources. Instances in Outpost subnets communicate with other instances in the AWS Region using private IP addresses, all within the same VPC.
- The following diagram illustrates the AWS Region us-west-2, two of its Availability Zones, and an Outpost. The VPC spans the Availability Zones and the Outpost. The Outpost is in an on-premises customer data center. Each zone in the VPC has one subnet, and each subnet has an instance.



This enables new classes of compute-intensive, latency sensitive applications latency. For example, a fleet of autonomous cars interacting with road sensors to prevent crashes, smart industrial robots assessing and reacting to plant conditions in a dangerous manufacturing environment, or retailers serving personalized promotions to shoppers' mobile phones in real time as they pass product displays.

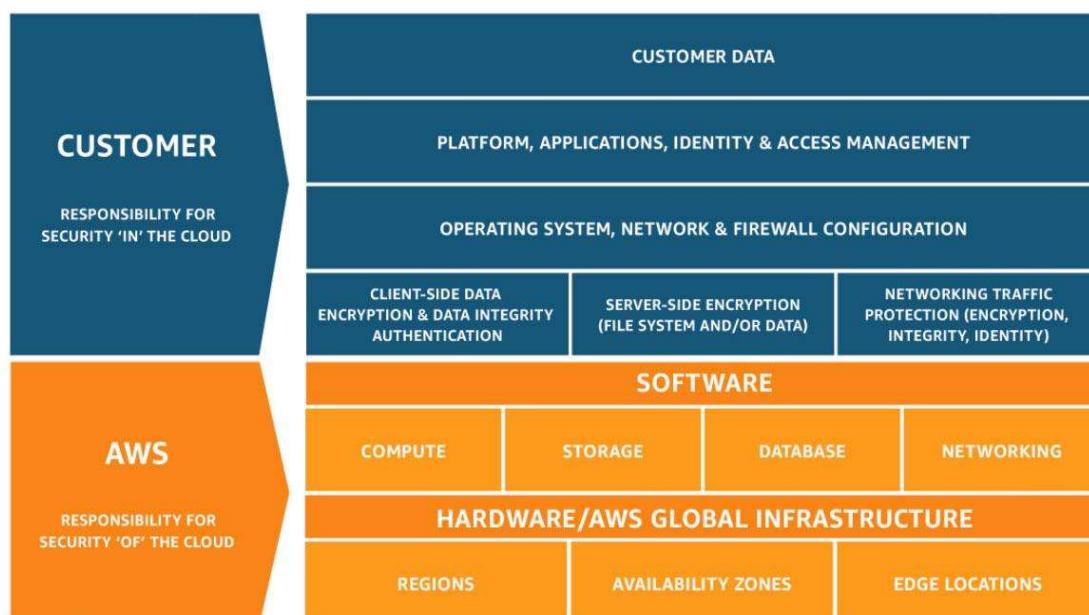
AWS for the Edge



AWS edge computing infrastructure and service platforms. Source: AWS

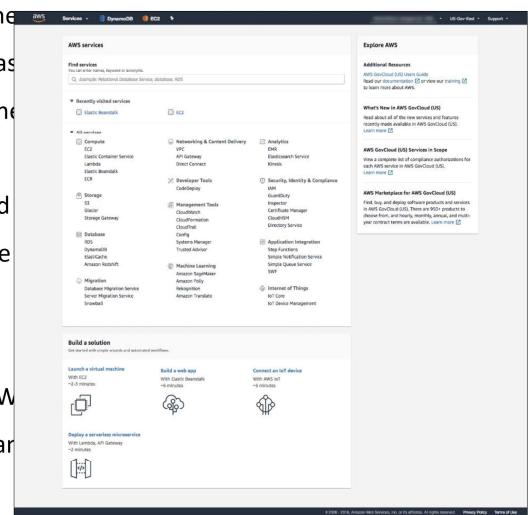
BITS Pilani

AWS & Customer



Using AWS - Connecting

- **AWS Management Console:** is a web application for managing AWS Cloud services. The console provides an intuitive user interface for performing many tasks. Each service has its own console, which can be accessed from the AWS Management Console. The console also provides information about the account and billing.
- **AWS Command Line Interface (CLI)** is a unified tool used to manage AWS Cloud services. With just one tool to download and configure, you can control multiple services from the command line and automate them through scripts.
- **The AWS Software Development Kits (SDKs)** provide an application programming interface (API) that interacts with the web services that fundamentally make up the AWS platform. The SDKs provide support for many different programming languages and platforms to allow you to work with your preferred language.



Using AWS - Video

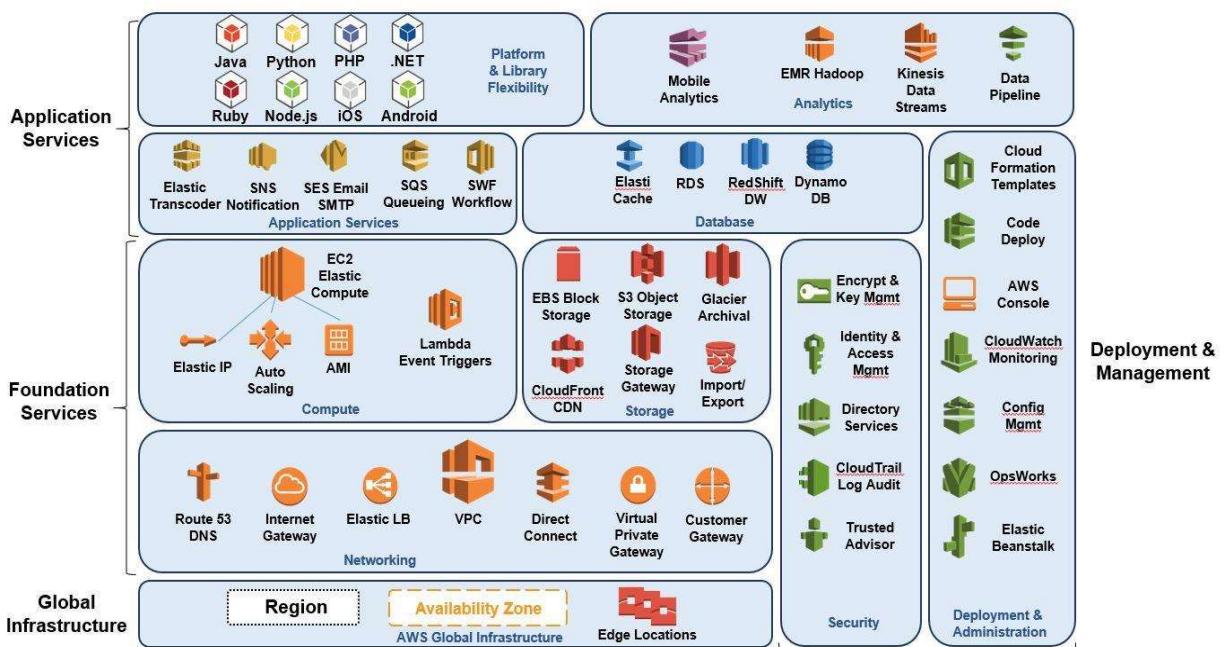
AWS Management Console: is a web application for managing AWS Cloud services.

The console provides an intuitive user interface for performing many tasks.

Each service has its own console, which can be accessed from the AWS Management Console.

The console also provides information about the account and billing.

AWS Reference Model



BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



AWS Elastic Compute Cloud EC2



- Virtual Compute Cloud
- Root-level System Access
- Elastic Capacity
- Management API
- Scale in Minutes
- Multiple Instance Sizes
- Network Security Model

EC2 Introduction

- Amazon EC2 is **AWS primary web service** that provides **resizable compute capacity** in the **cloud**.
 - Compute** refers to the amount of **computational power required to fulfill your workload**.
 - Amazon EC2 allows you to acquire compute through the **launching of virtual servers** called **instances**.
 - When you launch an **instance**, you can make use of the compute as you wish, just as you would with an on-premises server.
 - Users pay for the **computing power** of the instance. Charged per hour while the instance is running.
- When you **stop the instance**, you are **no longer charged**.



Amazon EC2

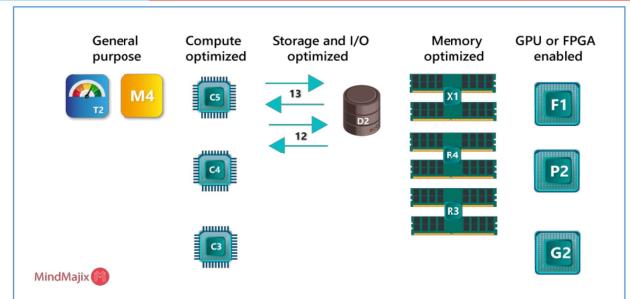


- Virtual Compute Cloud
- Root-level System Access
- Elastic Capacity
- Management API
- Scale in Minutes
- Multiple Instance Sizes
- Network Security Model

BITS Pilani

EC2 - Instance Type & AMI

- There are two concepts that are key to launching instances on AWS:
- (1) **Instance Type**: The amount of virtual hardware dedicated to the instance and
- (2) **AMI**: The software loaded on the instance.
 - AMI → Amazon Machine Image
- Note
 - Instance Type is similar to the processor**
 - AMI is similar to the OS**



aws EC2 instance types										
	General Purpose		Compute Optimized		Memory Optimized		Accelerated Computing		Storage Optimized	
Type	t2	m5	c5	r4	x1e	p3	h1	i3	d2	
Description	Burstable, good for changing workloads	Balanced, good for consistent workloads	High ratio of Compute to memory	Good for in-memory databases	Good for full in-memory applications	Good for graphics processing and other GPU uses	HDD backed, balance of compute and memory	SDD backed, balance of compute and memory	Highest disk ratio	
Mnemonic	t is for tiny or turbo	m is for main or happy medium	c is for compute	r is for RAM	x is for extreme	p is for pictures	h is for HDD	i is for IOPS	d is for dense	

ParkMyCloud

EC2 - Instance Type

- The **instance type** defines the **virtual hardware** supporting an Amazon EC2 instance.
- There are dozens of instance types available, varying in the following dimensions:
 - Virtual CPUs (vCPUs)
 - Memory
 - Storage (size and type)
- Network performance **Instance types** are grouped into **families** based on the **ratio of these values to each other**.
- For instance, the **m4 family** provides a balance of **compute, memory and network** resources, and it is a good choice for many applications.
- Within each family there are several choices that scale up linearly in size.
- Note that the ratio of vCPUs to memory is constant as the sizes scale linearly.

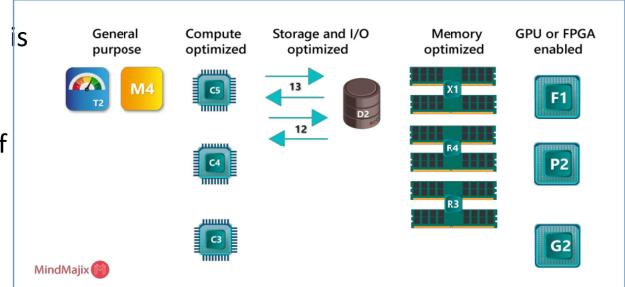
Instance Family	Instance Type(s)
General Purpose (M3)	M3.medium, M3.large, M3.xlarge, M3.2xlarge
Compute Optimized (C3)	C3.large, C3.xlarge, C3.2xlarge, C3.4xlarge, C3.8xlarge
Memory Optimized (R3)	R3.large, R3.xlarge, R3.2xlarge, R3.4xlarge, R3.8xlarge
Storage Optimized (I2, HS1)	I2.xlarge, I2.2xlarge, I2.4xlarge, I2.8xlarge, HS1.8xlarge
GPU (G2)	G2.2xlarge
Micro (T1, M1)	T1.micro, M1.small

Instance type	EBS only	NVME EBS	Instance store	Placement group	Enhanced networking
M6i	Yes	Yes	No	Yes	ENA EFA
M6id	No	Yes	NVMe	Yes	ENA EFA
M6idn	No	Yes	NVMe	Yes	ENA EFA
M6in	Yes	Yes	No	Yes	ENA EFA
M7a	Yes	Yes	No	Yes	ENA EFA
M7g	Yes	Yes	No	Yes	ENA EFA
M7gd	No	Yes	NVMe	Yes	ENA EFA
M7i	Yes	Yes	No	Yes	ENA EFA
M7i-flex	Yes	Yes	No	Yes	ENA
Mac1	Yes	Yes	No	Yes	ENA
Mac2	Yes	Yes	No	Yes	ENA
T2	Yes	No	No	Yes	Not supported
T3	Yes	Yes	No	Yes	ENA
T3a	Yes	Yes	No	Yes	ENA
T4g	Yes	Yes	No	Yes	ENA

BITS Pilani

EC2 - Instance Type

- Another variable to consider when choosing an instance type is **network performance**.
- For most instance types, AWS publishes a relative measure of **network performance**: *low, moderate, or high*.
- Some instance types specify a network performance of **10 Gbps**.
- The network performance increases within a family as the instance type grows.
- For workloads which require low latency, AWS provides enhanced networking support.

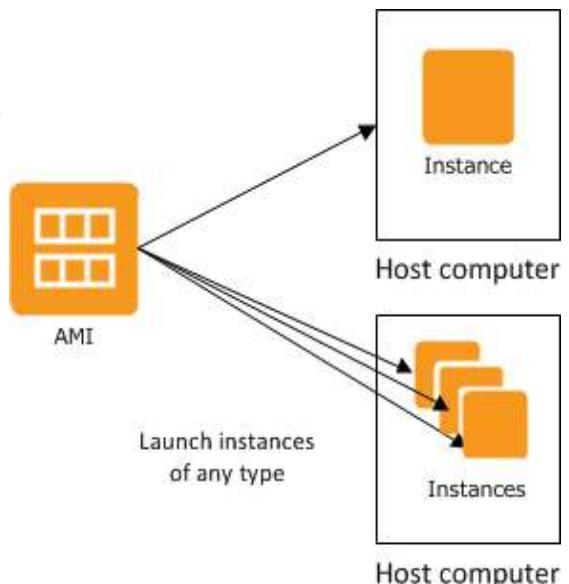


aws EC2 instance types										
	General Purpose		Compute Optimized		Memory Optimized		Accelerated Computing		Storage Optimized	
Type	t2	m5	c5	r4	x1e	p3	h1	i3	d2	
Description	Burstable, good for changing workloads	Balanced, good for consistent workloads	High ratio of Compute to memory	Good for in-memory databases	Good for full in-memory applications	Good for graphics processing and other GPU uses	HDD backed, balance of compute and memory	SDD backed, balance of compute and memory	Highest disk ratio	
Mnemonic	t is for tiny or turbo	m is for main or happy medium	c is for compute	r is for RAM	x is for extreme	p is for pictures	h is for HDD	i is for IOPS	d is for dense	

ParkMyCloud

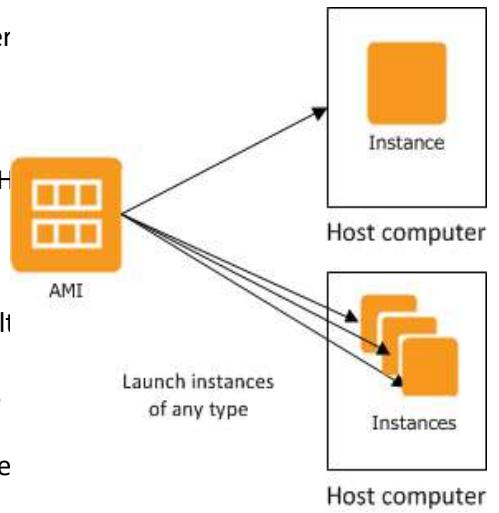
EC2 - Amazon Machine Image (AMI)

- The **Amazon Machine Image (AMI)** defines the **initial software** that will be on an instance when it is launched.
- An **AMI defines** every aspect of the **software state at instance launch**, including:
 - The Operating System (OS) and its configuration
 - The initial state of any patches
 - Application or system software
- All AMIs are based on x86 OSs, either Linux or Windows.



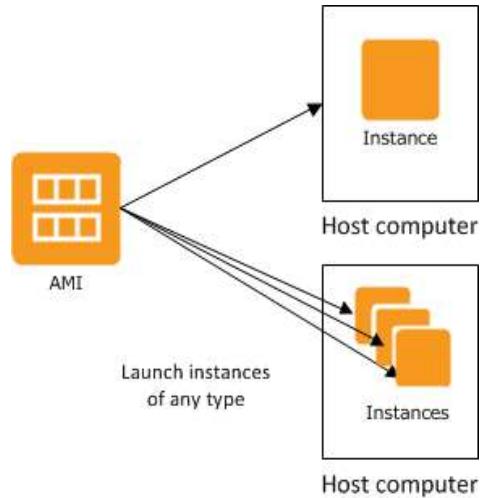
EC2 - AMI Types

- Published by AWS**— AWS publishes AMIs with versions of many different OSs, both Linux and Windows.
- These include multiple distributions of Linux (including Ubuntu, Red Hat, and Amazon's own distribution) and Windows 2008 and Windows 2012.
- Launching an instance based on one of these AMIs will result in the default OS settings, similar to installing an OS from the standard OS ISO image. As with any OS installation, you should immediately apply all appropriate patches upon launch.



EC2 - AMI Types

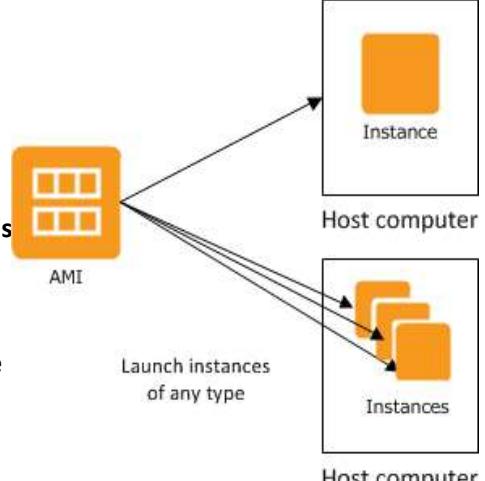
- **The AWS Marketplace**— AWS Marketplace is an **online store** that helps customers **find, buy, and immediately start using the software and services that run on Amazon EC2**.
- Many AWS partners have made their software available in the AWS Marketplace.
- This provides two benefits:
 - The customer **does not need to install** the software, and
 - The **license agreement is appropriate** for the cloud.
- Instances launched from an AWS Marketplace AMI incur the standard hourly cost of the instance type plus an additional per-hour charge for the additional software (some open-source AWS Marketplace packages have no additional software charge).



BITS Pilani

EC2 - AMI Types

- **Generated from Existing Instances**— An AMI can be created from an **existing Amazon EC2 instance**.
- This is a very **common source** of AMIs. Customers **launch an instance** from a published AMI, and then the **instance is configured to meet all the customer's corporate standards** for updates, management, security, and so on.
- An AMI is then generated from the configured instance and used to **generate instances** of that OS.
- In this way, all new instances **follow the corporate standard** and it is more difficult for individual projects to launch non-conforming instances.



BITS Pilani

EC2 - AMI Types

- **Uploaded Virtual Servers**— Using AWS VM Import/ Export service, customers can create images from various virtualization formats, including raw, VHD, VMDK, and OVA. The current list of supported OSs (Linux and Windows) can be found in the AWS documentation. It is incumbent on the customers to remain compliant.
- **VMDK** → Virtual Machine Disk : is a file format that describes containers for virtual hard disk drives to be used in virtual machines like VMware Workstation or VirtualBox.
- **VHD** → Virtual Hard Disk: is a file format which represents a **virtual hard disk** drive (HDD). It may contain what is found on a physical HDD, such as disk partitions and a file system, which in turn can contain files and folders. It is typically used as the hard disk of a virtual machine.
- **OVA** → Open Virtual Appliance/Application Format: is merely a single **file** distribution of the same **file** package, stored in the TAR format

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



Connecting to EC2 Instance



- Virtual Compute Cloud
- Root-level System Access
- Elastic Capacity
- Management API
- Scale in Minutes
- Multiple Instance Sizes
- Network Security Model

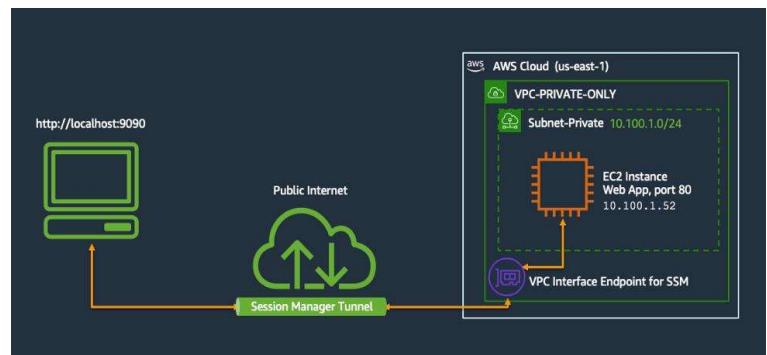
Creating an EC2 Instance - Video

- Amazon EC2 uses **public-key cryptography** to encrypt and decrypt login information.
- **Public-key cryptography** uses a **public key** to **encrypt** a piece of data and an **associated private key** to decrypt the data.
- These two keys together are called a **key pair**.
- **Key pairs** can be created through the **AWS Management Console**, CLI, or API, or customers can upload their own key pairs.
- **AWS stores the public key**, and the **private key** is kept by the **customer**.
- The **private key** is essential to acquiring **secure access** to an **instance** for the **first time**.

BITS Pilani

EC2 – Accessing over Web

- There are several ways that an instance may be addressed over the web upon creation:
- **Public Domain Name System (DNS) Name**— When you launch an instance, AWS creates a DNS name that can be used to access the instance. This DNS name is generated automatically and cannot be specified by the customer.
- **Public IP**— A launched instance may also have a public IP address assigned. This IP address is assigned from the addresses reserved by AWS and cannot be specified. This IP address is unique on the Internet, persists only while the instance is running, and cannot be transferred to another instance.
- **Elastic IP**— An elastic IP address is a static address unique on the Internet that you reserve independently and associate with an Amazon EC2 instance.

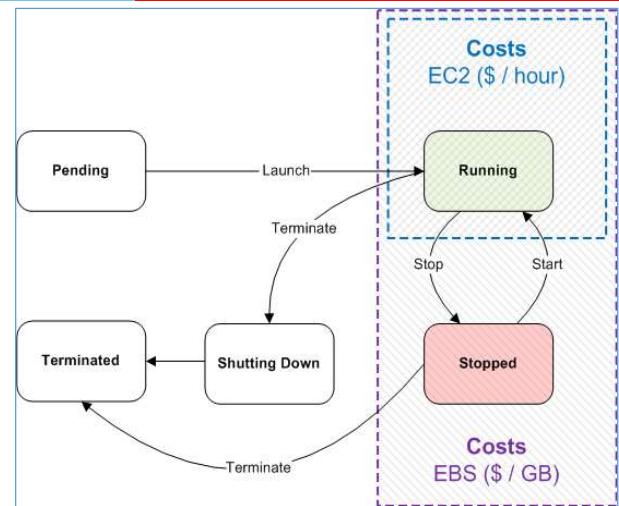


EC2 – Lifecycle

- Amazon EC2 has several features and services that facilitate the management of Amazon EC2 instances over their entire lifecycle.
- Launching
- Bootstrapping: The process of providing code to be run on an instance at launch is called bootstrapping.

Managing Instances

- When the number of instances in your account starts to climb, it can become difficult to keep track of them.
- Tags can help you manage not just your Amazon EC2 instances, but also many of your AWS Cloud services.
- Tags are key/ value pairs you can associate with your instance or other service.
- Tags can be used to identify attributes of an instance like project, environment (dev, test, and so on), billable department, and so forth.
- You can apply up to 10 tags per instance.



Monitoring Instances

AWS offers a service called Amazon CloudWatch that provides monitoring and alerting for Amazon EC2 instances, and other AWS infrastructure.

EC2 – Tenancy Options

- There are several tenancy options for Amazon EC2 instances that can help customers achieve security and compliance goals.
- Shared Tenancy** Shared tenancy is the default tenancy model for all Amazon EC2 instances, regardless of instance type, pricing model, and so forth. **Shared tenancy means that a single host machine may house instances from different customers.** As AWS does not use overprovisioning and fully isolates instances from other instances on the same host, this is a secure tenancy model.
- Dedicated Instances** Dedicated Instances run on hardware that's dedicated to a single customer. As a customer runs more Dedicated Instances, more underlying hardware may be dedicated to their account. Other instances in the account (those not designated as dedicated) will run on shared tenancy and will be isolated at the hardware level from the Dedicated
- Dedicated Host** An Amazon EC2 Dedicated Host is a physical server with Amazon EC2 instance capacity fully dedicated to a single customer's use. Dedicated Hosts can help you address licensing requirements and reduce costs by allowing you to use your existing server-bound software licenses.

EC2 – Placement Groups

- **Placement Groups** A placement group is a logical grouping of instances within a **single Availability Zone**. Placement groups enable **applications to participate** in a low-latency, **10 Gbps network**. To fully use this network performance for your placement group, choose an **instance type** that supports **enhanced networking** and 10 Gbps network performance.
- **Instance Stores** An instance store (sometimes referred to as **ephemeral storage**) provides **temporary block-level storage** for your instance. This storage is located on disks that are physically attached to the host computer.
- The size and type of **instance stores** available with an Amazon EC2 instance depend on the **instance type**. Can range from **no instance store** to **24 2 TB** instance store
- **Instance stores** are included in the cost of an **Amazon EC2 instance**, so they are a very cost-effective solution for appropriate workloads. The key aspect of instance stores is that they are temporary.
- Data in the instance store is lost when:
 - The underlying disk drive fails.
 - The instance stops (the data will persist if an instance reboots).
 - The instance terminates.

BITS Pilani

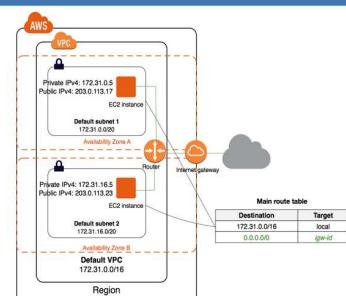


BITS Pilani

Pilani | Dubai | Goa | Hyderabad



Networking in AWS - VPC



AWS VPC



- A **virtual private cloud** (VPC) is a virtual network dedicated to your AWS account. It is **logically isolated** from other virtual networks in the AWS cloud.
- You can **launch your AWS resources**, such as Amazon EC2 instances, into your VPC. You can provision your own **logically isolated section of AWS**, similar to designing and implementing a separate **independent network** that would operate in an on-premises data center.
- You can configure your VPC; you can select its **IP address range**, **create subnets**, and **configure route tables**, **network gateways**, and **security settings**. A **subnet** is a **range of IP addresses** in your VPC. You can **launch AWS resources into a subnet that you select**.
- Use a **public subnet** for resources that must be connected to the Internet, and a **private subnet** for resources that won't be connected to the Internet. Within a **region**, you can create **multiple Amazon VPCs**, and each **Amazon VPC** is **logically isolated** even if it shares its IP address space..

uses

- ❖ Build virtual networks on the cloud
- ❖ No need for any VPN, hardware or physical DC
- ❖ Define bespoke network space like:
 - ❖ VPC with a single public subnet only
 - ❖ VPC with public and private subnets
 - ❖ VPC with public and private subnets and AWS Site-to-Site VPN access
 - ❖ VPC with a private subnet only and AWS Site-to-Site VPN access

BITS Pilani

AWS VPC - Components



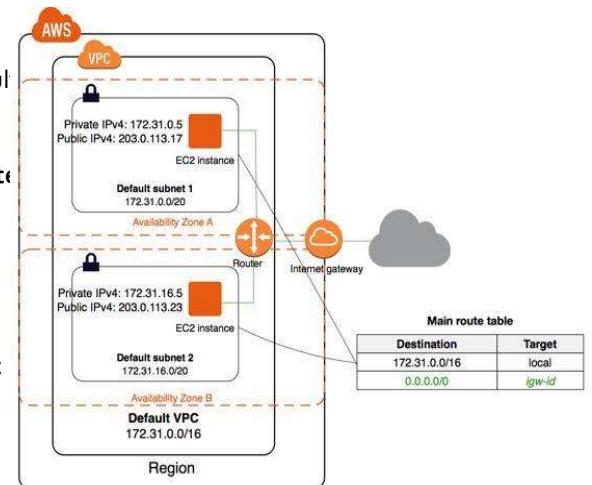
- An Amazon VPC consists of the following components:
 - Subnets
 - Route tables
 - Dynamic Host Configuration Protocol (DHCP) option sets
 - Security groups
 - Network Access Control Lists (ACLs)
- An Amazon VPC has the following optional components:
 - Internet Gateways (IGWs)
 - Elastic IP (EIP) addresses
 - Elastic Network Interfaces (ENIs)
 - Endpoints
 - Peering
 - Network Address Translation (NATs) instances and
 - NAT gateways Virtual Private Gateway (VPG), Customer Gateways (CGWs), and Virtual Private Networks (VPNs)



AWS VPC - Functioning



- You control how the instances that you launch into a VPC access resources outside the VPC.
- Your default VPC includes an Internet gateway, and each default subnet is a public subnet.
- Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address.
- These instances can communicate with the Internet through the Internet gateway.
- By default, each instance that you launch into a non-default subnet has a private IPv4 address, but no public IPv4 address, unless you specifically assign one at launch, or you modify the subnet's public IP address attribute.
- These instances can communicate with each other, but can't access the Internet.



BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

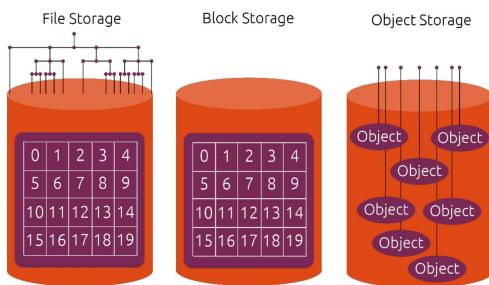


AWS Storage



Storage Types

- Block storage** : Operates at a lower level—the raw storage device level—and manages data as a set of numbered, fixed-size blocks.
- File storage** : Operates at a higher level—the operating system level—and manages data as a named hierarchy of files and folders.
- Block and file storage are often accessed over a network in the form of a Storage Area Network (SAN) for block storage, using protocols such as iSCSI or Fiber Channel, or as a Network Attached Storage (NAS) file server or “filer” for file storage.*



	Amazon Simple Storage Service (Amazon S3)	A service that provides scalable and highly durable object storage in the cloud.
	Amazon Glacier	A service that provides low-cost highly durable archive storage in the cloud.
	Amazon Elastic File System (Amazon EFS)	A service that provides scalable network file storage for Amazon EC2 instances.
	Amazon Elastic Block Store (Amazon EBS)	A service that provides block storage volumes for Amazon EC2 instances.
	Amazon EC2 Instance Storage	Temporary block storage volumes for Amazon EC2 instances.
	AWS Storage Gateway	An on-premises storage appliance that integrates with cloud storage.
	AWS Snowball	A service that transports large amounts of data to and from the cloud.
	Amazon CloudFront	A service that provides a global content delivery network (CDN).

BITS Pilani



AWS Simple Storage Service S3



AWS S3



- ❖ Amazon S3 is easy-to-use object storage with a simple web service interface that you can use to store and retrieve any amount of data from anywhere on the web.
- ❖ Amazon S3 also allows you to pay only for the storage you actually use, which eliminates the capacity planning and capacity constraints associated with traditional storage.
- ❖ Amazon S3 can be used alone or in conjunction with other AWS services, and it offers a very high level of integration with many other AWS cloud services.

uses

- ❖ Backup and archive for on-premises or cloud data
- ❖ Content, media, and software storage and distribution
- ❖ Big data analytics
- ❖ Static website hosting
- ❖ Cloud-native mobile and Internet application hosting
- ❖ Disaster recovery

SIEMENS
Ingenuity for life

Nasdaq

GE Healthcare

RYANAIR

nielsen

BITS Pilani

AWS S3



- ✓ Amazon S3 is cloud object storage. Instead of being closely associated with a server, Amazon S3 storage is independent of a server and accessed over the Internet.
- ✓ Instead of managing data as blocks or files using SCSI, CIFS, or NFS protocols, data is managed as objects using an Application Program Interface (API) built on standard HTTP verbs. Each Amazon S3 object contains both data and metadata.
- ✓ Objects reside in containers called buckets, and each object is identified by a unique user-specified key (filename).
- ✓ Buckets are a simple flat folder with no file system hierarchy.
- ✓ That is, you can have multiple buckets, but you can't have a sub-bucket within a bucket. Each bucket can hold an unlimited number of objects.
- ❖ However, keep in mind that **Amazon S3** is not a **traditional file system** and differs in significant ways.
- ❖ In **Amazon S3**, you **GET** an object or **PUT** an object, operating on the **whole object** at once, instead of **incrementally updating** portions of the object as you would with a file.
- ❖ Instead of a **file system**, **Amazon S3** is **highly-durable** and **highly-scalable object storage** that is optimized for reads and is built with an intentionally minimalistic feature set.
- ❖ It provides a **simple and robust abstraction** for file storage that frees you from many underlying details that you normally do have to deal with in traditional storage.
- ❖ Amazon S3 objects are automatically replicated on multiple devices in multiple facilities within a region.



AWS S3



Objects:

Opaque data to be stored (1 byte ... 5 Gigabytes)

Authentication and access controls

Buckets:

Object container – any number of objects

100 buckets per account

Keys:

Unique object identifier within bucket

Up to 1024 bytes long

Flat object storage model

Standards-Based Interfaces:

REST and SOAP

URL-Addressability – every object has a URL

Service:

ListAllMyBuckets

Buckets:

CreateBucket DeleteBucket

ListBucket GetBucketAccessControlPolicy

SetBucketAccessControlPolicy

GetBucketLoggingStatus SetBucketLoggingStatus

Objects:

PutObject PutObjectInline

GetObject GetObjectExtended

DeleteObject GetObjectAccessControlPolicy

SetObjectAccessControlPolicy

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



AWS Cloud Storage Services

AWS Elastic Block Storage EBS



AWS EBS



- ❖ Amazon EBS provides persistent block-level storage volumes for use with Amazon EC2 instances.
- ❖ Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability.
- ❖ Amazon EBS volumes are available in a variety of types that differ in performance characteristics and price.
- ❖ Multiple Amazon EBS volumes can be attached to a single Amazon EC2 instance, although a volume can only be attached to a single instance at a time.

uses

- ❖ Boot Volumes
- ❖ SQL & NoSQL Database
- ❖ Big Data workloads
- ❖ Data Warehouses
- ❖ Logging & Telemetry
- ❖ Transaction Processing



BITS Pilani

AWS EBS Types

General-Purpose SSD General-purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads. They deliver strong performance at a moderate price point that is suitable for a wide range of workloads.



A general-purpose SSD volume can range in size from 1 GB to 16 TB and provides a baseline performance of three IOPS per gigabyte provisioned, capping at 10,000 IOPS.

They are suited for a wide range of workloads where the very highest disk performance is not critical, such as:

- System boot volumes
- Small- to medium-sized databases
- Development and test environments

AWS EBS Types



Provisioned IOPS SSD Provisioned IOPS SSD volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads that are sensitive to storage performance and consistency in random access I/O throughput. While they are the most expensive Amazon EBS volume type per gigabyte, they provide the highest performance of any Amazon EBS volume type in a predictable manner.

A Provisioned IOPS SSD volume can range in size from 4 GB to 16 TB. Provisioned IOPS SSD volumes provide predictable, high performance and are well suited for:

- Critical business applications that require sustained IOPS performance
- Large database workloads

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



AWS Elastic File Storage EFS



AWS EFS



- ❖ EFS(Elastic file system) is a file-level storage service that basically provides a shared elastic file system with virtually unlimited scalability support
- ❖ EFS is highly available storage that can be utilized by many servers at the same time. AWS EFS is a fully managed service by amazon and it offers scalability on the fly.
- ❖ This means that the user need not worry about their increasing or decreasing workload. If the workload suddenly becomes higher then storage will automatically scale itself and if the workload decreases then the storage will itself scale down.
- ❖ This scalability feature of EFS also provides cost benefits as you need not pay anything for the part of storage that you don't use, you only pay for what you use(Utility-based computing).

uses

- ❖ Lift-and-shift application support
- ❖ Analytics for big data
- ❖ Web server support
- ❖ Application development and testing



BITS Pilani

AWS EFS vs AWS EBS vs AWS S3

Category	S3	EBS	EFS
Storage Type	Object Storage	Block Storage	File Storage
Pricing	Pay as you Use	Pay for provisioned capacity	Pay as you Use
Storage Size	Unlimited Storage	Limited storage	Unlimited Storage
Scalability	Unlimited Scalability	Increase/decrease size manually	Unlimited Scalability
Durability	Stored redundantly across multiple Azs	Stored redundantly in a Single AZ	Stored redundantly across multiple Azs
Availability	Max is 99.99% with S3	99.9%	No SLAs
Security	Supports Data at Rest and Data in Transit encryption	Supports Data at Rest and Data in Transit encryption	Supports Data at Rest and Data in Transit encryption
Back up and Restore	Use Versioning or cross-region replication	Automated Backups and Snapshots	EFS to EFS replication
Performance	Slower than EBS and EFS	Faster than S3 and EFS	Faster than S3, Slower than EBS
Accessibility	Publicly and Privately accessible	Accessible only via the attached EC2 instance	Accessible simultaneously from multiple EC2 and on-premises instances
Interface	Web Interface	File System Interface	Web and File System Interface
Use cases	Media, Entertainment, Big data analytics, backups and archives, web serving and content management	Boot volumes, transactional and NoSQL databases, data warehousing ETL	Media, Entertainment, Big data analytics, backups and archives, web serving and content management, home directories



AWS Glacier



Amazon Glacier



- Amazon Glacier is an extremely low-cost storage service that provides durable, secure, and flexible storage for data archiving and online backup.
- To keep costs low, Amazon Glacier is designed for infrequently accessed data where a retrieval time of three to five hours is acceptable.
- Amazon Glacier can store an unlimited amount of virtually any kind of data, in any format.
- In most cases, the data stored in Amazon Glacier consists of large **TAR** (Tape Archive) or ZIP files.
- In **Amazon Glacier**, data is stored in **archives**. An **archive** can contain up to **40TB** of data, and you can have an **unlimited number of archives**.
- Each **archive** is assigned a unique archive ID at the time of creation.
- (Unlike an Amazon S3 object key, you cannot specify a user-friendly archive name.) All archives are automatically **encrypted**, and **archives are immutable**— after an archive is created, it cannot be modified.

uses

- ❖ Digital Storage.
- ❖ Scientific Data Storage.
- ❖ Healthcare information Archiving.
- ❖ Regulatory and Compliance Archiving.
- ❖ Magnetic Tape Replacement.



Rock & Roll Hall of Fame preserves rock music history and modernizes on AWS.

[Read the case study »](#)



Qube Cinema cuts costs by 80% with archival on Amazon S3 Glacier.

[Read the case study »](#)



Reuters builds easily accessible large-scale news archives on Amazon S3 Glacier.

[Read the blog »](#)



BandLab decreases costs and improves availability using Amazon S3 Glacier.

[Read the case study »](#)



joyn readies exclusive content for audiences with Amazon S3 Intelligent-Tiering and Amazon S3 Glacier.

[Read the blog »](#)



AWS Case Study



NETFLIX



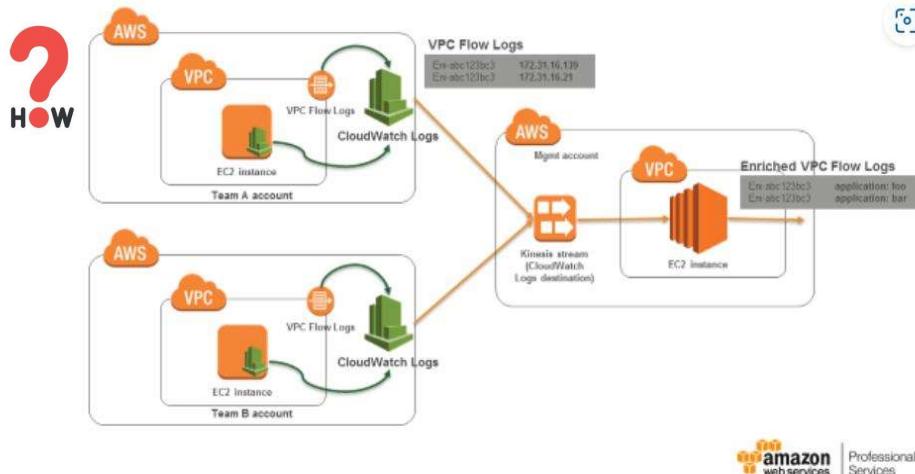
- Online content provider Netflix can support seamless global service by using Amazon Web Services (AWS). AWS enables Netflix to quickly deploy thousands of servers and terabytes of storage within minutes. Users can stream Netflix shows and movies from anywhere in the world, including on the web, on tablets, or on mobile devices such as iPhones.

[AWS re:Invent 2022 - Capacity plan optimally in the cloud \(NFX304\) - YouTube](#)

Amazon Web Services (AWS) offers hundreds of different types of Amazon Elastic Cloud Compute (EC2) options, and Netflix has a multitude of different stateful and stateless workloads that need different combinations of CPU, RAM, disk, and network.

In this 2022 AWS re:Invent session, learn how Netflix uses a service capacity modeling system to optimally consume Amazon EC2 to run a variety of uncertain workloads ranging from Cassandra databases to stateless Java applications.

NETFLIX



What happens when you need to move 89 million viewers to a different AWS region? Netflix's infrastructure, built on AWS, makes it possible to be extremely resilient, even when the company is running services in many AWS Regions simultaneously. In this episode of This is My Architecture, Coburn Watson, director of performance and reliability engineering at Netflix, walks through the company's DNS architecture—built on Amazon Route 53 and augmented with Netflix's Zuul—that allows the team to evacuate an entire region in less than 40 minutes.

[Netflix: Multi-Regional Resiliency and Amazon Route 53 - YouTube](#)

BITS Pilani



Agenda



- ❖ AWS Recap
- ❖ OpenStack Introduction
 - ❖ What is OpenStack
 - ❖ OS Reference Model
 - ❖ Introducing Network Function Virtualization (NFV)
 - ❖ OS Case Study

What is AWS



Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.



Global Infrastructure: AWS serves over one million active customers in more than 190 countries, and it continues to expand its global infrastructure



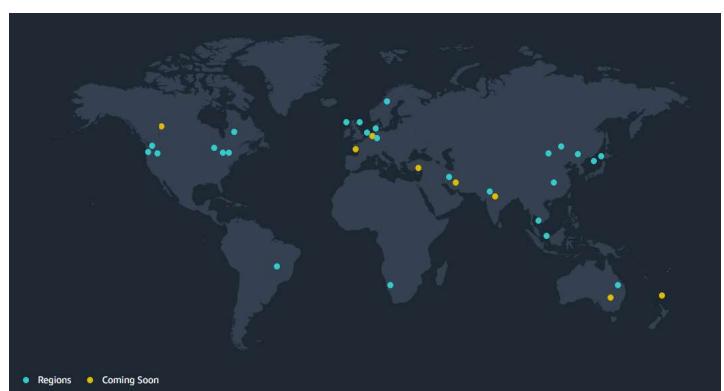
Security: All AWS customers benefit from data center and network architectures built to satisfy the requirements of the most security-sensitive organizations.

- Application building blocks
- Stable APIs
- Proven Amazon infrastructure
- Focus on innovation and creativity
- Long-term investment

BITS Pilani

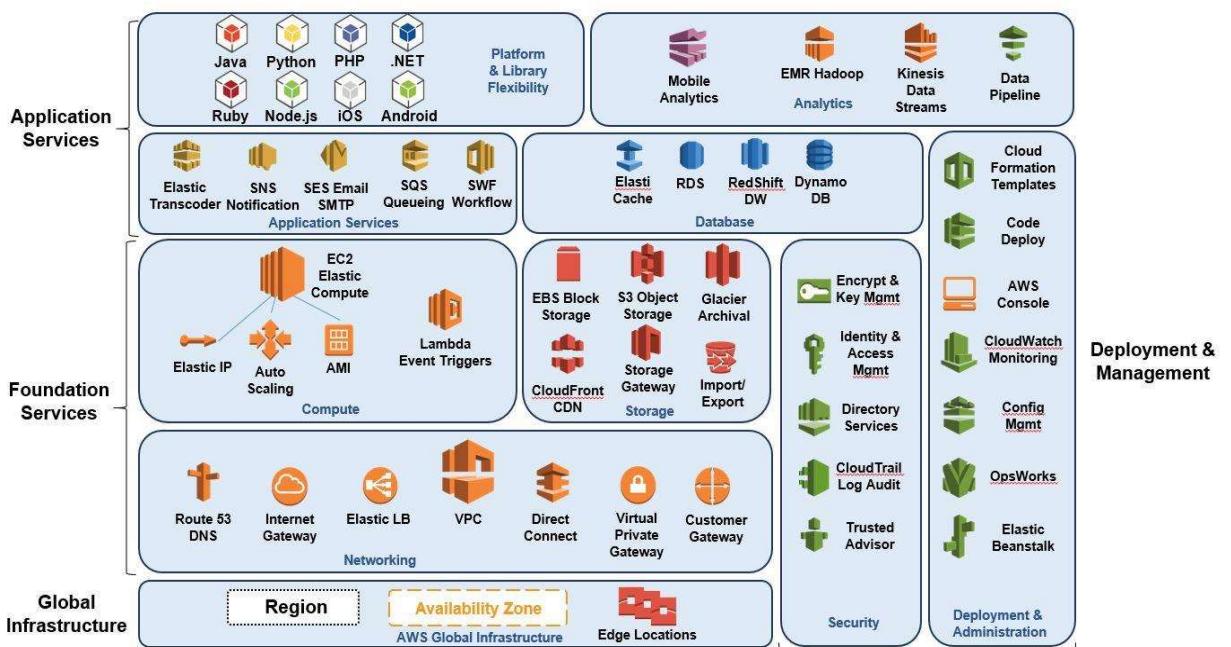
AWS Regions & Availability Zones

- AWS provides a **highly available technology infrastructure platform** with multiple **locations worldwide**. These locations are composed of **regions** and **Availability Zones**. Each **region** is a **separate geographic area**.
- Each **region** has **multiple, isolated locations** known as **Availability Zones**. AWS enables the **placement of resources** and data in multiple **locations**. Resources **aren't replicated** across regions **unless organizations** choose to do so.
- Additionally, for faster delivery of content, AWS has **EDGE locations** concentrated in major cities. (Used with CloudFront & Route53)



Availability Zones located in **AWS Regions** consist of one or more **discrete data centers**, each of which has **redundant power**, **networking**, and **connectivity**, and is housed in **separate facilities**. Each AZ has multiple internet connections and power connections to multiple grids.

AWS Reference Model

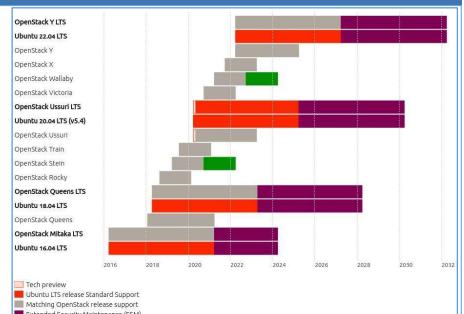


BITS Pilani

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

OpenStack



What is OpenStack?

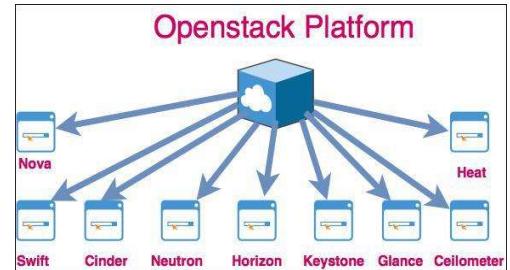
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources.

Managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds.

Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing.

Managed by OpenStack Foundation a non profit organization.



BITS Pilani

History of Open stack

OpenStack was created during the first months of 2010. Rackspace wanted to rewrite the infrastructure code running its Cloud servers offering, and considered open sourcing the existing Cloud files code. At the same time, Anso Labs (contracting for NASA) had published beta code for Nova, a Python-based “cloud computing fabric controller”.

Both efforts converged and formed the base for OpenStack. The first Design Summit was held in Austin, TX on July 13-14, 2010, and the project was officially announced at OSCON in Portland, OR, on July 21st, 2010.



Mission Statement

The Four Opens

| << | >> |

Open Source

We do **not** produce "open core" software.

We are committed to creating truly open source software that is usable and scalable. Truly open source software is not feature or performance limited. There will be no "Enterprise Edition".

We use the Apache License, 2.0.

- QSI approved
- GPLv3 compatible
- DFSG compatible

Open Design

We are committed to an **open design process**. Every development cycle the OpenStack community holds face-to-face events to gather requirements and write specifications for the upcoming release. Those events, which are **open to anyone**, include users, developers, and upstream projects. We gather requirements, define priorities and flesh out technical design to guide development for the next development cycle.

The community controls the design process. You can help make this software meet your needs.

Open Development

We maintain a publicly available source code repository through the entire development process. We do public code reviews. We have public roadmaps. This makes participation simpler, allows users to follow the development process and participate in QA at an early stage.

Open Community

One of our core goals is to maintain a healthy, vibrant developer and user community. Most decisions are made using a [lazy consensus](#) model. All processes are documented, open and transparent.

The technical governance of the project is provided by the community itself, with contributors electing team leads and members of the Technical Committee.

All project meetings are held in public IRC channels and recorded. Additional technical communication is through public mailing lists and is archived.



BITS Pilani

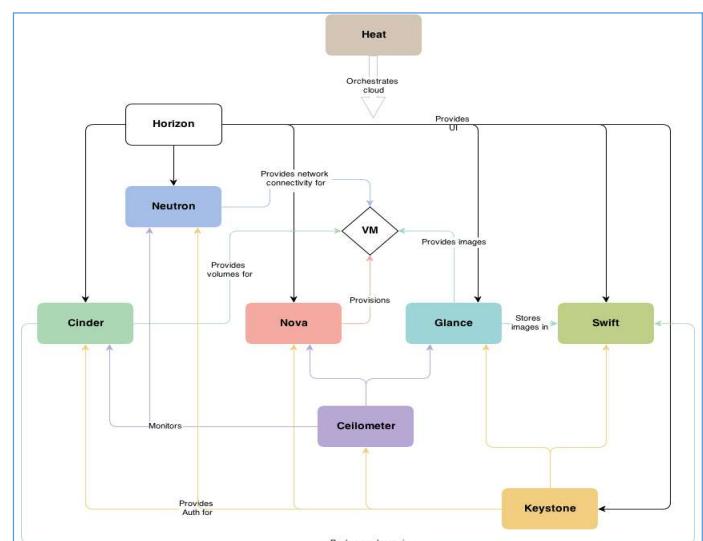
What you get with OS

OpenStack is an [IaaS cloud computing project](#) that is free open-source software.

Providing infrastructure means that OpenStack makes it easy for users to quickly add new instance, upon which other cloud components can run. Typically, the infrastructure then runs a "platform" upon which a developer can create software applications that are delivered to the end users.

The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center.

Users either manage it through a web-based dashboard, through [command-line tools](#), or through a [RESTful Application Programming Interface](#) (API). OpenStack.org released it under the terms of the [Apache License](#).



OpenStack Projects

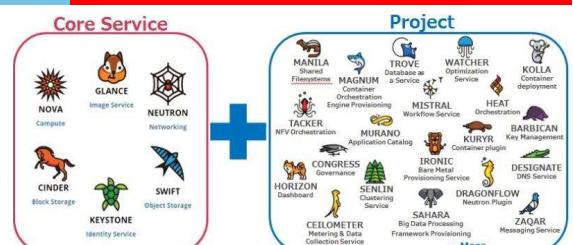
- The OpenStack platform is actually composed of multiple components, called **projects**.
- Each project is managed by a **technical committee** and the **OpenStack Foundation** decides which projects are ready to be included in the **OpenStack core**.
- These **projects work together** to provide the **services required** to deliver the Cloud.
- OpenStack has the following set of resources available to setup the cloud infrastructure:
 - Compute Resources
 - Network Resources
 - Block Storage
 - Identity Management
 - New projects are being added with each release and as the OpenStack community calls for them. New projects underway include metering, application orchestration, and database-as-a-service

Edition	Release name	Release date	Component
1	Austin	21-10-2010	Nova, Swift
2	Bexar	03-02-2011	Nova, Glance, Swift
5	Essex	05-04-2012	Nova, Glance, Swift, Horizon, Keystone
6	Folsom	27-09-2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
7	Havana	17-10-2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
8	Icehouse	17-04-2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove
9	Juno	16-10-2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara
14	Newton	06-10-2016	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqr, Manila, Designate, Barbican, Searchlight, Magnum, aodh, cloudkitty, congress, freezer, mistral, monasca-api, monasca-log-api, murano, panko, senlin, solum, tacker, vitrage, watcher

BITS Pilani

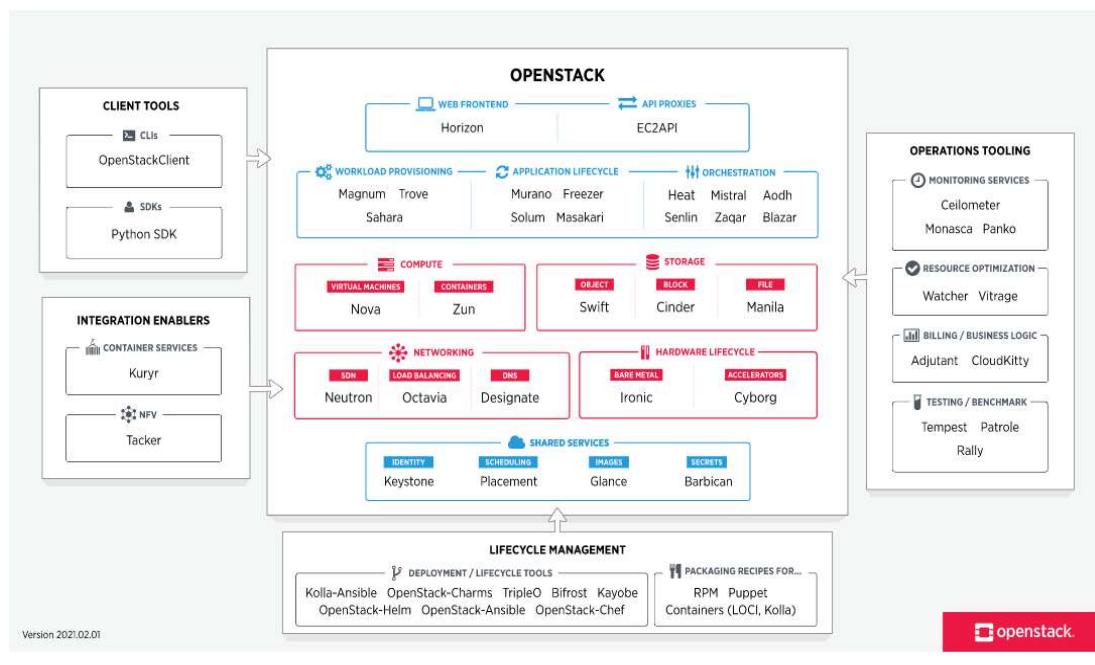
OpenStack Projects

Service	Project name	Description
Dashboard	Horizon	Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of virtual machines on demand.
Networking	Neutron	Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Storage		
Object Storage	Swift	Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.
Block Storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Shared services		
Identity service	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
Image Service	Glance	Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
Higher-level services		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Database Service	Trove	Provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.



As of 2023, the latest release, maintained by OpenStack is 2023.1 Antelope.

OpenStack Reference Model



BITS Pilani

OpenStack Projects

Nova

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

Swift

Swift service providing object storage which allows users to store and retrieve files. Swift architecture is distributed to allow for horizontal scaling, and to provide redundancy as failure-proofing. Data replication is managed by software, allowing greater scalability and redundancy than dedicated hardware.

Glance

Glance service that acts as a registry for virtual machine images to allowing users to copy server images for immediate storage.

Images can be used as templates when setting up new servers. Usually the images are stored in the Swift (Object) service.

OpenStack Services

An OpenStack deployment contains a number of components providing APIs to access infrastructure resources. This page lists the various services that can be deployed to provide such resources to cloud end users.

Compute	Shared Services	Workload Provisioning
NOVA Compute Service	KEYSTONE Identity service	
ZUN Containers Service	PLACEMENT Placement service	
Hardware Lifecycle	Glance	MAGNUM Container Orchestration Engine Provisioning
Ironic Bare Metal Provisioning Service	BARBICAN Key management	SAHARA Big Data Processing Framework Provisioning
CYBORG Lifecycle management of accelerators		TROVE Database as a Service
Storage	Orchestration	Application Lifecycle
SWIFT Object store	HEAT Orchestration	MASAKARI Instances High Availability Service
CINDER Block Storage	SENLIN Clustering service	MURANO Application Catalog
MANILA Shared filesystems	MISTRAL Workflow service	SOLUM Software Development Lifecycle Automation
Networking	ZAQAR Messaging Service	FREEZER Backup, Restore, and Disaster Recovery
NEUTRON Networking	BLAZAR Resource reservation service	EC2API EC2 API proxy
OCTAVIA Load balancer	AODH Alarming Service	API Proxies
DESIGNATE DNS service	MAGNUM Container Orchestration Engine Provisioning	Web frontends
	SAHARA Big Data Processing Framework Provisioning	HORIZON Dashboard
Workload Provisioning	TROVE Database as a Service	SKYLINE Next generation dashboard (tech preview)
Application Lifecycle		
	MASAKARI Instances High Availability Service	
	MURANO Application Catalog	
	SOLUM Software Development Lifecycle Automation	
	FREEZER Backup, Restore, and Disaster Recovery	

BITS Pilani

OpenStack Services

Tooling: Those services deliver APIs primarily targeted to cloud admins and deployers, to help with cloud operations.

Software in this section facilitates integration of OpenStack components in adjacent open infrastructure stacks

Monitoring services

CEILOMETER	Metering & Data Collection Service
PANKO	Event, Metadata Indexing Service
MONASCA	Monitoring

Resource optimization

WATCHER	Optimization Service
VITRAGE	Root Cause Analysis service

Billing / Business Logic

ADJUTANT	Operations processes automation
CLOUDKITTY	Billing and chargebacks

Testing / Benchmark

RALLY	Benchmarking tool
TEMPEST	The OpenStack Integration Test Suite
PATROLLE	The OpenStack RBAC Integration Test Suite

Swift add-ons

STORLETS	Computable object storage
----------	---------------------------

Integration enablers

Software in this section facilitates integration of OpenStack components in adjacent open infrastructure stacks.

Containers

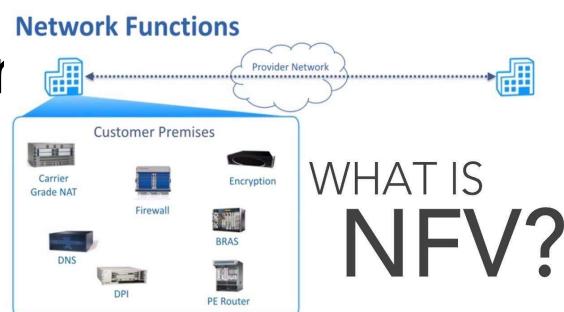
KURYR	OpenStack Networking integration for containers
-------	---

NFV

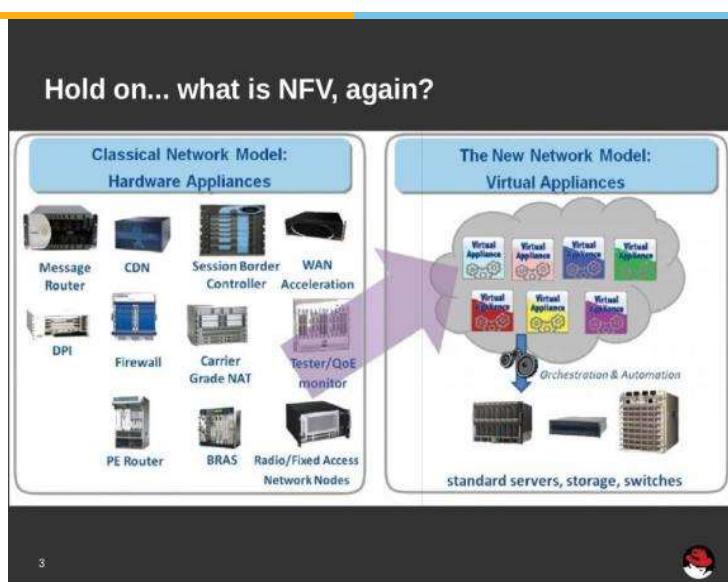
TACKER	NFV Orchestration
--------	-------------------



Introducing Network Function Virtualization - NFV



What is NFV?



Network functions virtualization (NFV) is a way to [virtualize network services](#), such as routers, firewalls, and load balancers, that have traditionally been run on proprietary hardware. These services are packaged as [virtual machines \(VMs\)](#) on commodity hardware, which allows service providers to run their network on standard servers instead of proprietary ones.

It is one of the primary components of a [telco cloud](#), which is reshaping the telecommunications industry.

NFV improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources.

NFV – Origins & Motivations

The traditional phone network and perhaps even telegram networks are examples of the earliest data transport networks. Early on, the design criteria and quality benchmark by which networks were judged were latency, availability, throughput, and the capacity to carry data with minimal loss.

These factors directly influenced the development and requirements for the hardware and equipment to transport the data (text and voice, in this case). Additionally, hardware systems were built for very specific use cases and targeted functions, ran tightly coupled proprietary operating systems on them, and were meant to perform only specific functions.

With the advent of data transport networks, the requirements and factors that influence the network's design and the devices' efficiency stayed unchanged (for example, the network design should achieve highest throughput with minimum latency and jitter over extended distances with minimal loss).

Separate Appliance for each Function
Proprietary Software: Designed to Run on Custom Hardware
Proprietary Hardware: Custom FPGA/ASIC/Optics/CPU ...
Fixed Network Function
Limited Scalability: Physical Space and Power Limitations

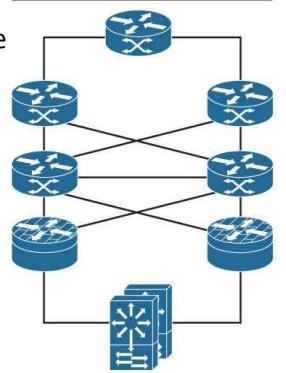


Figure 1.1 Traditional Network Devices

BITS Pilani

NFV – Origins & Motivations

All the traditional networking devices were made for specific functions, and the data networks built were tailored and customized to meet these efficiency criteria effectively. The software or code running on these custom-designed hardware systems was tightly coupled to it, closely integrated with the silicon Field Programmable and Customized Integrated Circuits and focused exclusively on performing the specific functions of the device.

Separate Appliance for each Function
Proprietary Software: Designed to Run on Custom Hardware
Proprietary Hardware: Custom FPGA/ASIC/Optics/CPU ...
Fixed Network Function
Limited Scalability: Physical Space and Power Limitations

The NFV architecture proposed by the [European Telecommunications Standards Institute \(ETSI\)](#) is helping to define standards for NFV implementation. Each component of the architecture is based on these standards to promote better stability and interoperability.

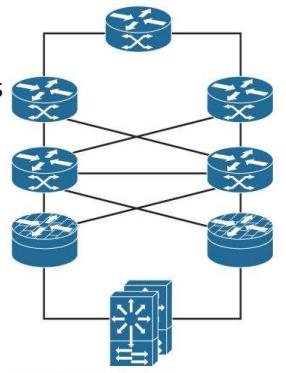


Figure 1.1 Traditional Network Devices

What is NFV

- Network Functions Virtualization, NFV is an approach to telecommunications networking where the network entities that traditionally used dedicated hardware items are now replaced with computers on which software runs to provide the same functionality.
- By running a network based around NFV, Network Functions Virtualization techniques, it is easier to expand and modify the network, and it is able to provide considerably more flexibility as well as being able to standardize on much of the hardware as it consists of additional computing power. In this way costs can be considerably reduced.
- NFV, network functions virtualization is a concept that virtualizes major elements of a network. In this way, rather than having a dedicated item of hardware to provide a given function, software running on a computer / server is used.

What is NFV

- In this way entire classes of network node functions can be set up as building blocks that can be connected to create overall telecommunications networks. With NFV, you don't need to have dedicated hardware for each network function.
- NFV utilizes traditional server virtualization, but extends the concept significantly. In this way one or more virtual machines running different software and providing different processes, on top of industry standard high volume servers, are able to provide the functions of switches and storage, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function.
- Examples of the virtualized functions that can be provided include: virtualized load balancers, firewalls, intrusion detection devices, WAN accelerators, routers, access control and billing.

NFV Architecture

NFV architecture consists of:

- **Virtualized network functions (VNFs)** are software applications that deliver network functions such as file sharing, directory services, and IP configuration.
- **Network functions virtualization infrastructure (NFVi)** consists of the infrastructure components—compute, storage, networking—on a platform to support software, such as a hypervisor like KVM or a container management platform, needed to run network apps.
- **Management, automation and network orchestration (MANO)** provides the framework for managing NFV infrastructure and provisioning new VNFs.

NFV Comparison

How Is NFV Different from Traditional Physical Devices

	NFV	Traditional physical devices
General-purpose hardware	Standard x86-based servers are used, as well as general-purpose storage devices and switching devices.	Dedicated devices are used.
Software-hardware decoupling	Software is decoupled from hardware and offered as modules.	Hardware and software are tightly coupled. Software functions depend on dedicated hardware.
Openness	A universal hardware platform and standard interfaces facilitate an open ecosystem through multi-party cooperation.	The use of dedicated devices leads to a closed system, making it difficult to introduce third-party partners.
Network flexibility	General-purpose hardware and resource virtualization technologies allow software and hardware resources to be dynamically adjusted to service requirements.	Dedicated devices do not support virtualization technologies for resource sharing and elastic scaling.
Upgrade convenience	Device upgrades are fast, as they mainly involve software.	Network device upgrades and deployment are time-consuming, because they require both software and hardware development.
Automated O&M	It virtualizes hardware resources and makes O&M more automatic and intelligent.	Device upgrade and replacement processes are complicated, because maintenance requires manual or semi-manual preparation and configuration through the CLI or web system.
Service association	NFV networks are deployed based on services, and can be flexibly orchestrated.	Traditional networks are relatively independent. Service requirements cannot be quickly converted to network requirements, and the network response is slow.

NFV Benefits

With NFV, service providers can run network functions on standard hardware instead of dedicated hardware.

Also, because network functions are virtualized, multiple functions can be run on a single server. This means that less physical hardware is needed, which allows for resource consolidation that results in physical space, power, and overall cost reductions.

NFV gives providers the flexibility to run VNFs across different servers or move them around as needed when demand changes.

This flexibility lets service providers deliver services and apps faster.

For example, if a customer requests a new network function, they can spin up a new VM to handle that request. If the function is no longer needed, the VM can be decommissioned.

This can also be a low-risk way to test the value of a potential new service.

BITS Pilani



BITS Pilani
Pilani | Dubai | Goa | Hyderabad



Comparing AWS & OpenStack

Overview

AWS and OpenStack are cloud computing giants that enjoy a vast customer base globally. Although AWS is more popular than OpenStack, the latter is catching up fast. As far as the question goes, “which is better?” it largely depends on your company’s specific requirements.

So, without further ado, let’s take a look at some of AWS and OpenStack’s core aspects so you can make an informed decision!

Differences

In principle, the main difference between AWS and OpenStack is that the former already exists, while the latter one you have to build yourself.

Since building cloud infrastructure from scratch entails significant upfront investments (setting up a data centre, hardware purchase, cloud deployment consulting fee, etc.), AWS is usually a more compelling option at the beginning of the cloud migration journey.

At the end of the day, all you need to do is to create an account on AWS and attach a credit card, and you can start using cloud resources right away.

AWS will charge you based on the actual resource consumption.

Differences

On the other hand, those costs(AWS) can quickly become significant as the number of workloads continues to increase.

Over time, it may turn out that the aggregated recurring cost of using AWS will exceed the cost of OpenStack deployment.

Of course, running cloud infrastructure on-premises also comes with some recurring costs (hosting facilities, power consumption, staff salary, etc.), but these are lower when running workloads in the long-term and at scale, according to [Canonical's Cloud Pricing Report from 2021](#).

BITS Pilani



BITS Pilani
Pilani | Dubai | Goa | Hyderabad



OpenStack Case Study



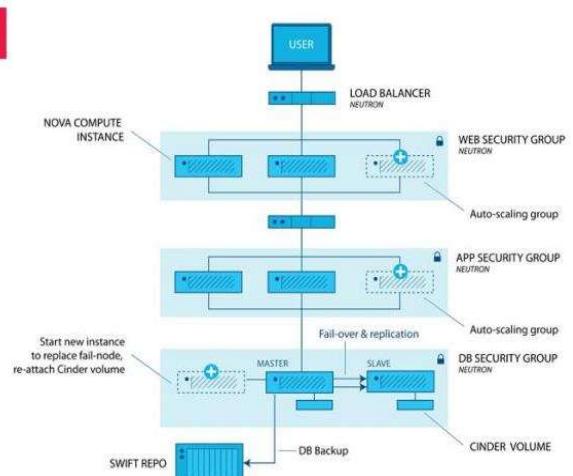
Generic 3-tier Web Application



Interactive web applications are the most prevalent applications in business today.

Consumers and enterprises alike interact with their employees, customers and partners online, using applications such as online banking, human resources and even tax filing and pet adoption. Many organizations, such as Workday, Betfair, Ancestry.com, JFE Steel, HMRC (Her Majesty's Revenue and Customs) and LivePerson are using OpenStack to deliver interactive web applications at scale.

IT resource needs for web applications often fluctuate with end user demand—predictably or unpredictably. Failure to respond to either can impact customer satisfaction and sales. The ability to dynamically add and remove resources is one of the primary benefits to using an OpenStack cloud.



BITS Pilani

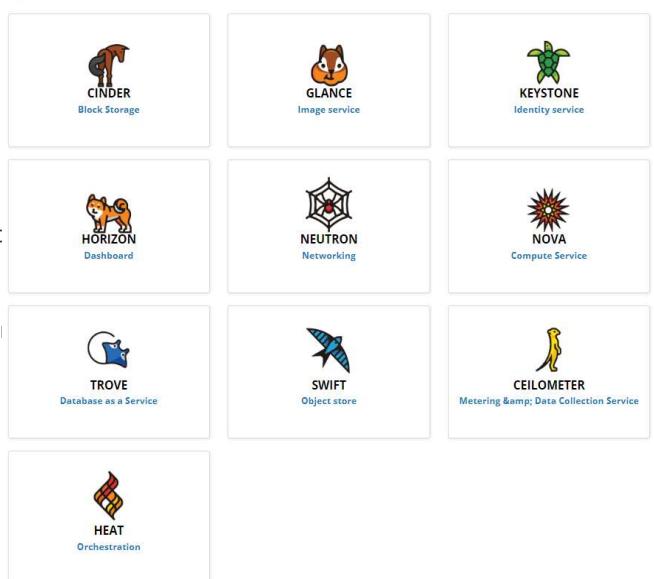
OpenStack Services Used

Read this [Web Applications reference architecture](#) to learn how to use OpenStack services to build a three-tier web application on an existing OpenStack cloud. And, try it yourself with the [Heat template](#) provided in the Community Application Catalog!

The Heat template sets up new virtual machines, private networks for each tier and the proper connections, load balancers, routers and security groups. It deploys the popular LAMP software into the tiers, and WordPress as an example web application. Two templates are provided in the package—one for auto-scaling and one for manual scaling

The target OpenStack cloud must be configured with the six core projects and Heat (in order to use the deployment template). Ceilometer is required for auto-scaling deployment. Trove is referenced but not used in the sample configuration.

OpenStack Services included in this configuration (10 of 65)



Agenda



- ❖ IaaS Recap
- ❖ VM Management
 - ❖ Why VM Management?
 - ❖ VM management Tools
 - ❖ VM Provisioning
 - ❖ VM Migration Techniques
 - ❖ Live Migration Example

2

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

What is AWS



Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.



Global Infrastructure: AWS serves over one million active customers in more than 190 countries, and it continues to expand its global infrastructure

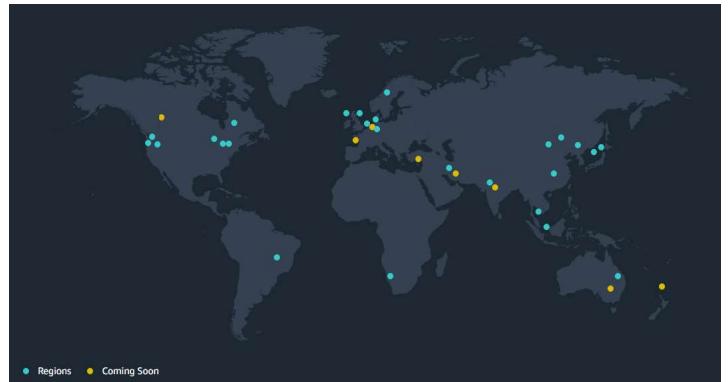


Security: All AWS customers benefit from data center and network architectures built to satisfy the requirements of the most security-sensitive organizations.

- Application building blocks
- Stable APIs
- Proven Amazon infrastructure
- Focus on innovation and creativity
- Long-term investment

AWS Regions & Availability Zones

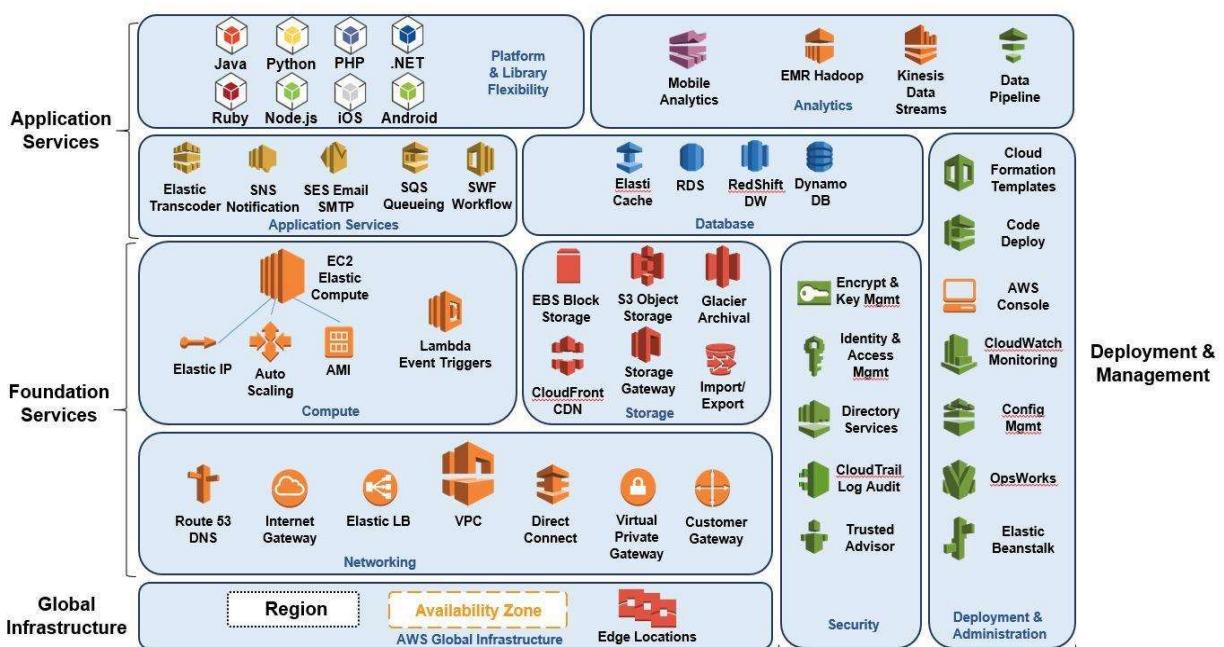
- AWS provides a **highly available technology infrastructure platform** with multiple **locations worldwide**. These locations are composed of **regions** and **Availability Zones**. Each **region** is a **separate geographic area**.
- Each **region** has **multiple, isolated locations** known as **Availability Zones**. AWS enables the **placement of resources** and data in multiple **locations**. Resources **aren't replicated** across regions **unless organizations** choose to do so.
- Additionally, for faster delivery of content, AWS has **EDGE locations** concentrated in major cities. (Used with CloudFront & Route53)



Availability Zones located in **AWS Regions** consist of one or more **discrete data centers**, each of which has **redundant power**, **networking**, and **connectivity**, and is housed in **separate facilities**. Each AZ has multiple internet connections and power connections to multiple grids.

BITS Pilani

AWS Reference Model



What is OpenStack?

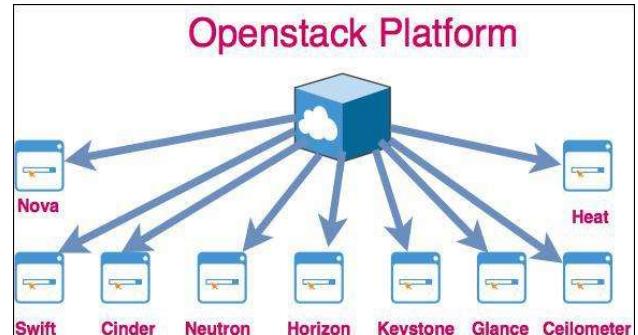
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources.

Managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds.

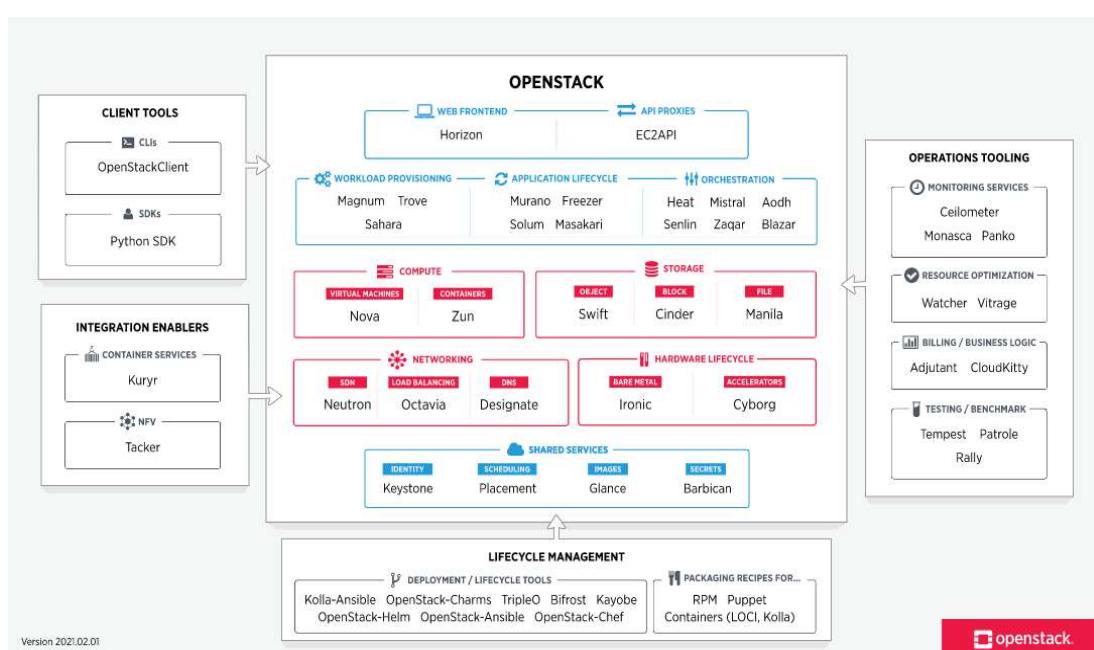
Backed by some of the biggest companies in software development and hosting, as well as thousands of individual community members, many think that OpenStack is the future of cloud computing.

Managed by OpenStack Foundation a non profit organization.



BITS Pilani

OpenStack Reference Model



openstack

Differences

In principle, the main difference between AWS and OpenStack is that the former already exists, while the latter one you have to build yourself.

Since building cloud infrastructure from scratch entails significant upfront investments (setting up a data centre, hardware purchase, cloud deployment consulting fee, etc.), AWS is usually a more compelling option at the beginning of the cloud migration journey.

At the end of the day, all you need to do is to create an account on AWS and attach a credit card, and you can start using cloud resources right away.

AWS will charge you based on the actual resource consumption.

BITS Pilani

Differences

On the other hand, those costs(AWS) can quickly become significant as the number of workloads continues to increase.

Over time, it may turn out that the aggregated recurring cost of using AWS will exceed the cost of OpenStack deployment.

Of course, running cloud infrastructure on-premises also comes with some recurring costs (hosting facilities, power consumption, staff salary, etc.), but these are lower when running workloads in the long-term and at scale, according to [Canonical's Cloud Pricing Report from 2021](#).

BITS Pilani



VM Management

Anatomy of Cloud

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

In traditional physical resources, machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.

In a **virtual infrastructure**, this configuration must be done **on-the-fly**, with as little time between the time the VMs are requested and the time they are available to the users.

This is further complicated by the need to **configure groups** of VMs that will provide **a specific service** (e.g., an application requiring a Web server and a database server).

Additionally, a **virtual infrastructure manager** must be capable of **allocating resources efficiently**, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and **reacting to changes in the physical infrastructure**.

Anatomy of Cloud

Virtual infrastructure management in **private clouds** has to deal with an additional problem:

Unlike large IaaS cloud providers such as Amazon, **private clouds typically do not have enough** resources to provide the illusion of "infinite capacity."

The **immediate provisioning scheme used in public clouds**, where resources are provisioned at the moment they are requested, is **ineffective in private clouds**.

Support for additional provisioning schemes is required for applications that require resources at specific times, such as **best-effort provisioning** and **advance reservations** to guarantee quality of service (QoS).

Thus, **efficient resource allocation algorithms** and **policies** and the ability to combine both **private and public cloud resources**, resulting in a hybrid approach, become even more important.

BITS Pilani

What do we manage in Public Cloud?

When we talk about **RM**, we are referring to the pillars of **IaaS**. The Amazon service provides hundreds of **pre-made AMIs** (Amazon Machine Images) with a variety of operating systems (i.e., Linux, Open Solaris, or Windows) and pre-loaded software.

- **Compute Resource**
 - **Storage Resource**
 - **Memory**
 - **Network**
- It provides you with complete control of your computing resources and lets you run on Amazon's computing and infrastructure environment easily.
 - It also reduces the time required for obtaining and booting a new server's instances to minutes, thereby allowing a quick scalable capacity and resources, up and down, as the computing requirements change.

Other ancillary services can be added as required. The challenge is that most cloud ecosystems are hybrid consisting of several VM's

Amazon offers different instances' size according to
(a) the resources' needs (small, large, and extra large),
(b) the high CPU's needs it provides (medium and extra large high CPU instances), and
(c) high-memory instances (extra large, double extra large, and quadruple extra large instance).

What do we manage in Private Cloud?

When we talk about RM, we are referring to the pillars of IaaS

- Compute Resource
- Storage Resource
- Memory
- Network

Other ancillary services can be added as required. The challenge is that most cloud ecosystems are hybrid consisting of several VM's

Private cloud exhibits a highly virtualized cloud data centre located inside your organization's firewall.

It may also be a private space dedicated for your company within a cloud vendor's data centre designed to handle the organization's workloads, and in this case it is called Virtual Private Cloud (VPC). Private clouds exhibit the following characteristics:

- 1) Allow service provisioning and compute capability for an organization's users in a self service manner.
- 2) Automate and provide well-managed virtualized environments.
- 3) Optimize computing resources, and servers' utilization.
- 3) Support specific workloads.

The best-known examples are Eucalyptus and OpenNebula

BITS Pilani

Why Resource Management

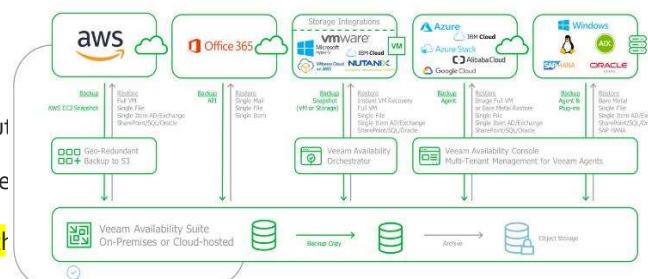
With the infrastructure world in constant flux, more and more businesses are adopting a multi-cloud deployment model.

Consider the impact on the data alone. 10 years ago, all anyone worried about was if the SAN would stay up, and if it didn't, would their data be protected

Fast forward to today, even a small business can have data scattered across the globe.

Maybe they have a few vSphere hosts in an HQ, with branch offices using workloads running in the cloud or Software as a Service-based applications.

Maybe backups are stored in an object storage repository (somewhere — but only one guy knows where). This is happening in the smallest of businesses, so as a business grows and scales, the challenges become even more complex.



Key Considerations

- **How to Protect my VM Instance**
- **How to Recover VM in case of Failure**
- **How to achieve smooth scaling**

What is Resource Management

RM is a process that deals with the procurement and release of resources.

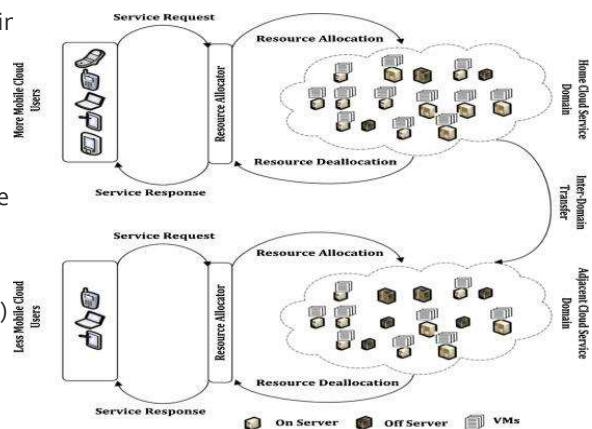
Virtualization techniques are used for flexible and on-demand resource provisioning.

To do so, for each received task, either a new VM is created or it is placed on the existing VM of the same user.

Once the task is completed, all the acquired resources are released which become parts of the free resource pool.

Resource assignment is performed on the basis of Service Level Agreement (SLA) that is agreed between the service provider and the customer.

SLA contains details of the service level that is required by a tenant. Moreover, it contains information about the payment process and SLA violation penalty.



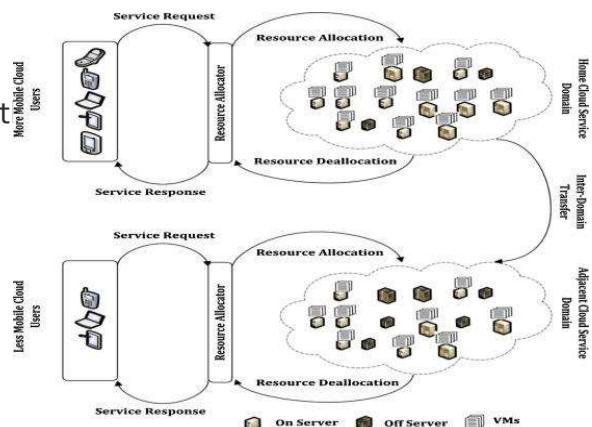
BITS Pilani

What is Resource Management

Cloud resource management requires complex policies and decisions for multi-objective optimization.

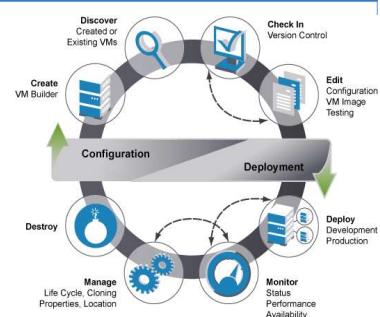
Effective resource management is extremely challenging due to the scale of the cloud infrastructure and to the unpredictable interactions of the system with a large population of users.

The scale makes it impossible to have accurate global state information and the large user population makes it nearly impossible to predict the type and the intensity of the system workload.





VM Lifecycle



VM Life Cycle

Virtual Machine Life Cycle

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

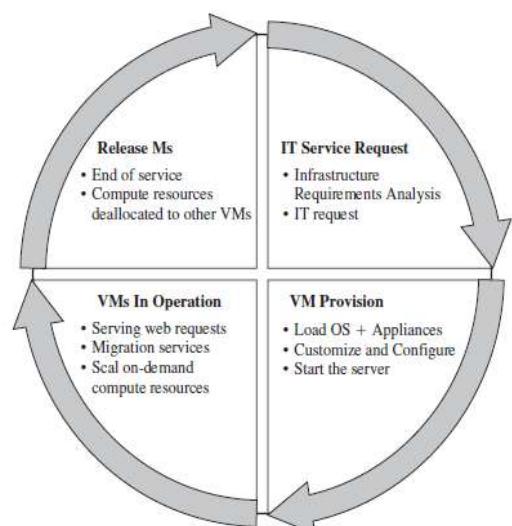


FIGURE 5.3. Virtual machine life cycle.

VM Provisioning process

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

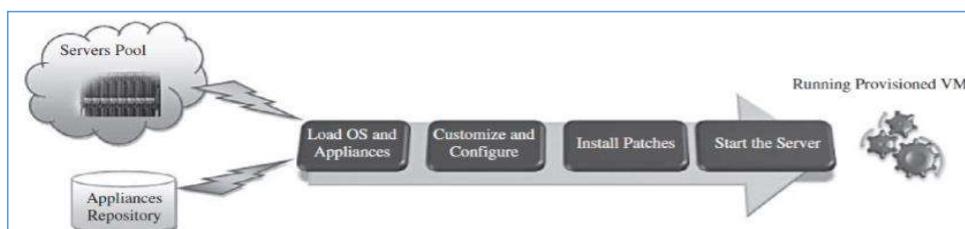
VMs can be provisioned by

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

VM Provisioning process

Steps to Provision VM -

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



VM Provisioning using templates

After creating a virtual machine by virtualizing a physical server, or by building a new virtual server in the virtual environment, a template can be created out of it.

Most virtualization management vendors (VMware, XenServer, etc.) provide the data center's administration with the ability to do such tasks

Provisioning from a template reduces the time required to create a new virtual machine.

- Administrators can create different templates for different purposes.

For example –

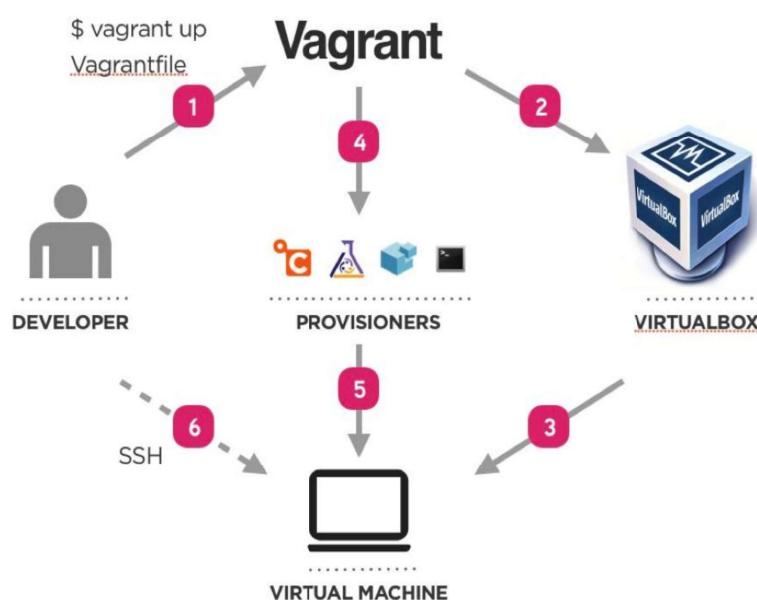
- Vagrant provision tool using VagrantFile (template file) - [Demo](#)
- Heat – Orchestration Tool of openstack (Heat template in YAML format) - [Demo](#)

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

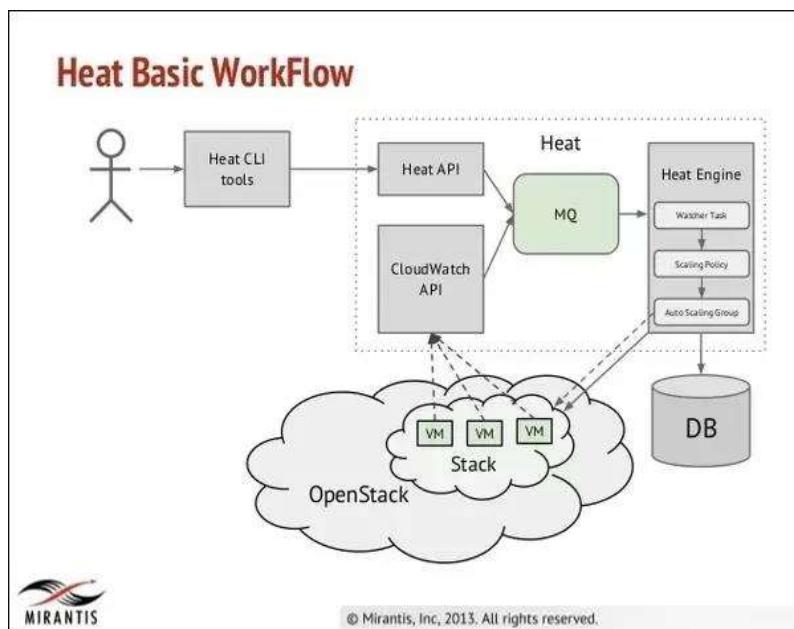
Provisioning from a template is an invaluable feature, because it reduces the time required to create a new virtual machine.

BITS Pilani

Understanding Vagrant



Demo – HEAT Provisioning



BITS Pilani

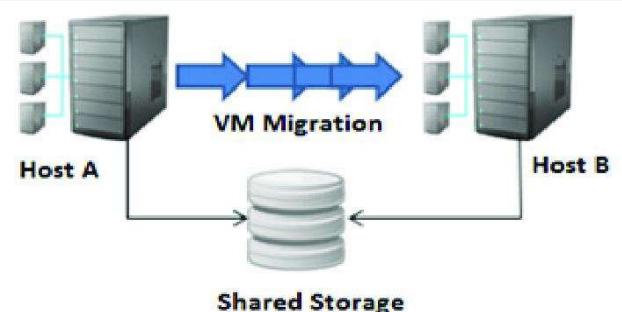


BITS Pilani

Pilani | Dubai | Goa | Hyderabad



VM Migration



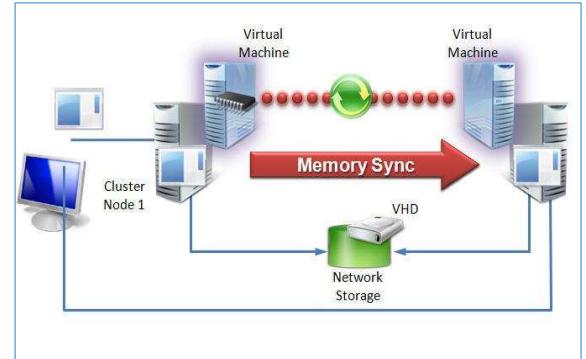
What is VM Migration

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

There are different techniques of VM migration-

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

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



What is VM Migration

- Migration can be categorized as cold or non-live migration and live migration.
- Based on granularity, the migration can be divided into single and multiple migrations.
- The design and continuous optimization and improvement of live migration mechanisms are striving to minimize downtime and live migration time.
- The downtime is the time interval during the migration service is unavailable due to the need for synchronization.
- For a single migration, the migration time refers to the time interval between the start of the pre-migration phase to the finish of post-migration phases that instance is running at the destination host.
- On the other hand, the total migration time of multiple migrations is the time interval between the start of the first migration and the completion of the last migration.

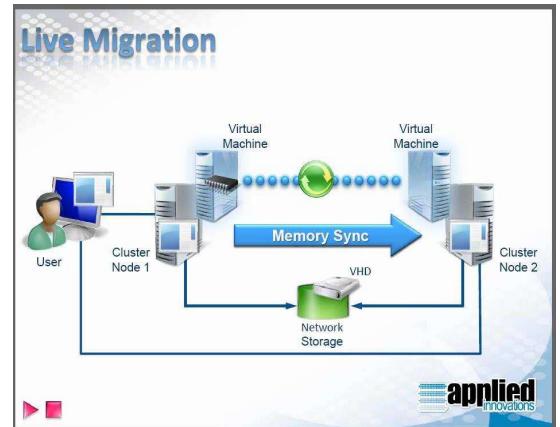
What is VM Live Migration

Live migration can be defined as the movement of a virtual machine from one physical host to another while being powered on.

When it is properly carried out, this process takes place without any noticeable effect from the end user's point of view (a matter of milliseconds).

One of the most significant advantages of live migration is the fact that it facilitates proactive maintenance in case of failure, because the potential problem can be resolved before the disruption of service occurs.

Live migration can also be used for load balancing in which work is shared among computers in order to optimize the utilization of available CPU resources.



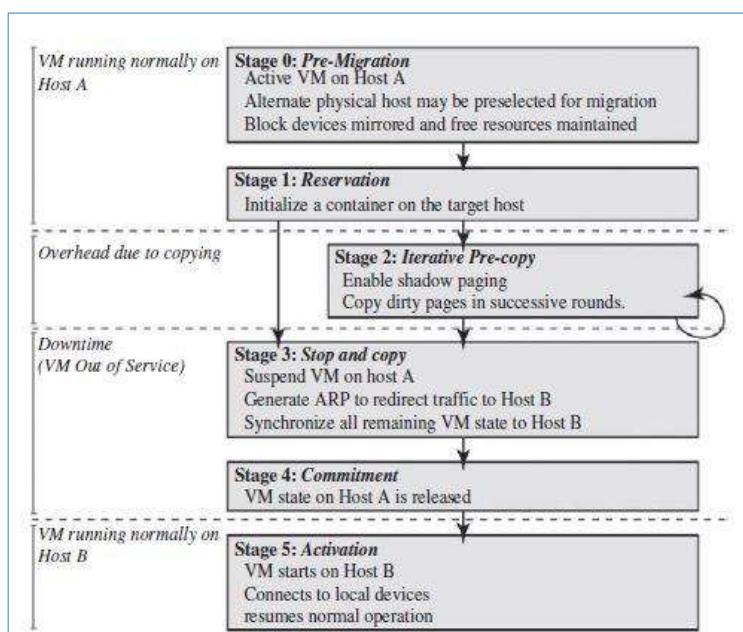
BITS Pilani

Live Migration Xen Hypervisor

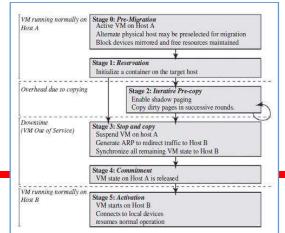
The steps of live migration's mechanism and how memory and virtual machine states are being transferred, through the network, from one host A to another host B:

The Xen hypervisor is an example for this mechanism.

The migration process has been viewed as a transactional interaction between the two hosts involved:



Live Migration Xen Hypervisor



Stage 0: Pre-Migration. An active virtual machine exists on the physical host A.

Stage 1: Reservation. A request is issued to migrate an OS from host A to host B (a precondition is that the necessary resources exist on B and on a VM container of that size).

Stage 2: Iterative Pre-Copy. During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.

Stage 3: Stop-and-Copy. Running OS instance at A is suspended, and its network traffic is redirected to B. CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage, there is a consistent suspended copy of the VM at both A and B. The copy at A is considered primary and is resumed in case of failure.

Stage 4: Commitment. Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as a commitment of the migration transaction. Host A may now discard the original VM, and host B becomes the primary host.

Stage 5: Activation. The migrated VM on B is now activated. Post-migration code runs to reattach the device's drivers to the new machine and advertise moved IP addresses.

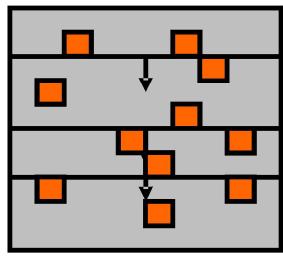
Live Migration Xen Hypervisor

Memory and storage transmission can be categorized into three phases:

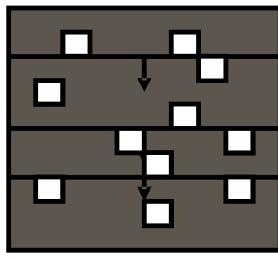
- Push phase where the instance is still running in the source host while memory pages and disk block or writing data are pushed through the network to the destination host.
- Stop-and-Copy phase where the instance is stopped, and the memory pages or disk data is copied to the destination across the network. At the end of the phase, the instance will resume at the destination.
- Pull phase where the new instance executes while pulling faulted memory pages when it is unavailable in the source from the source host across the network.

Live Migration Process

Pre-copy migration : Round 1, Enable Shadow Paging

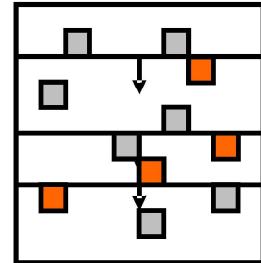


Host A

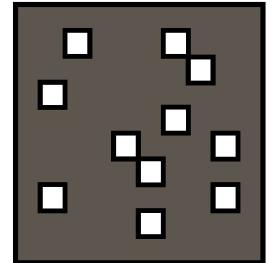


Host B

Pre-copy migration : Round 2

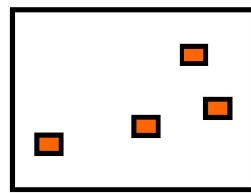


Host A



Host B

Stop & Copy – Final Round



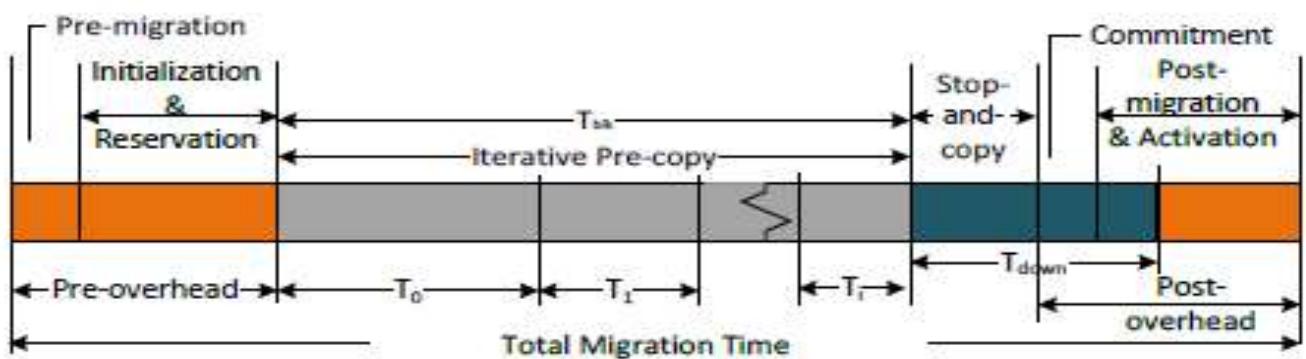
Host A



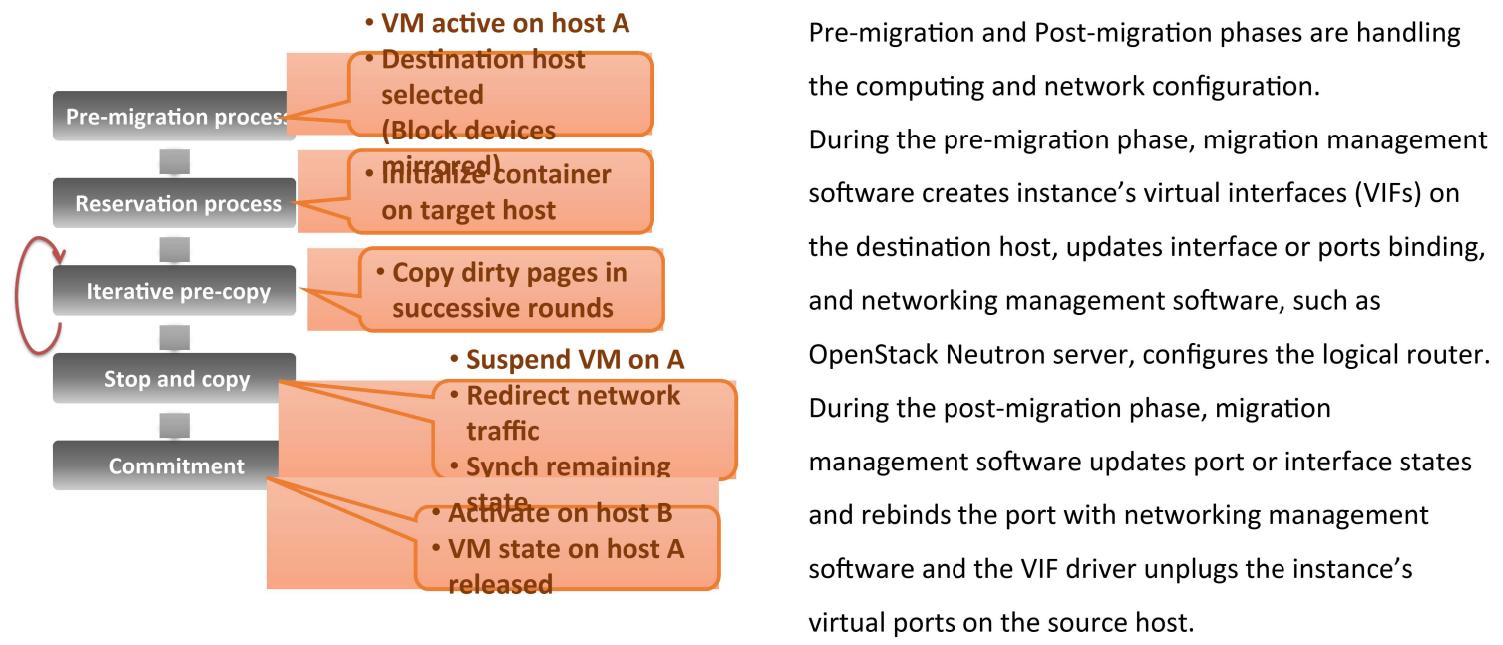
Host B

BITS Pilani

Live Migration Process



Live Migration Technique



BITS Pilani

Live Migration Technique

Post-migration code runs to reattach the device's drivers to the new machine and advertise moved IP addresses.

This approach to failure management ensures that at least one host has a consistent VM image at all times during migration:

- 1) Original host remains stable until migration commits and that the VM may be suspended and resumed on that host with no risk of failure.
- 2) A migration request essentially attempts to move the VM to a new host and on any sort of failure, execution is resumed locally, aborting the migration.

Challenges of live migration :

- VMs have lots of state in memory
- Some VMs have soft real-time requirements

Live Migration Effect on a Running Web Server

Clark et al. evaluated the mentioned migration on Apache 1.3 Web Server; that served a static content at a high rate. The throughput is achieved when continuously serving a single 512-KB file to a set of 100 concurrent clients.

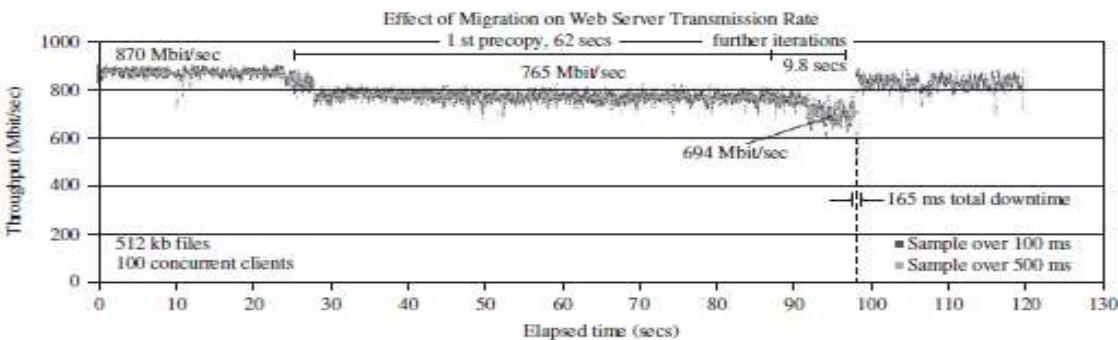


FIGURE 5.6. Results of migrating a running Web server VM [21].

Live Migration Vendor Implementations Example

There are lots of VM management and provisioning tools that provide the live migration of VM facility

VMware VMotion:

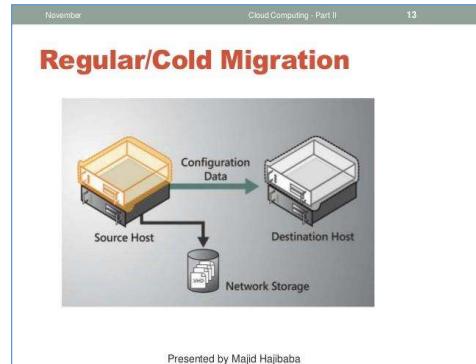
- Automatically optimize and allocate an entire pool of resources for maximum hardware utilization, flexibility, and availability.
- Perform hardware's maintenance without scheduled downtime along with migrating virtual machines away from failing or underperforming servers.

Citrix XenServer "XenMotion":

Based on Xen live migrate utility, it provides the IT Administrator the facility to move a running VM from one XenServer to another in the same pool without interrupting the service (hypothetically zero – downtime server maintenance), making it a highly available service and also good feature to balance workloads on the virtualized environments.

Cold Migration

- **Cold migration** is the migration of a powered-off virtual machine.
- With cold migration, you have the option of moving the associated disks from one data store to another. The virtual machines are not required to be on a shared storage.
- It's important to highlight that the two main differences between live migration and cold migration are that live migration needs a shared storage for virtual machines in the server's pool, but cold migration does not;
- Also, in live migration for a virtual machine between two hosts, there would be certain CPU compatibility checks to be applied; while in cold migration this checks do not apply.



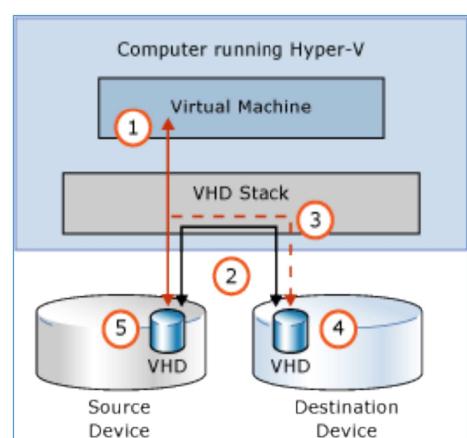
Note:

- The configuration files, including the NVRAM file (BIOS settings), log files, as well as the disks of the virtual machine, are moved from the source host to the destination host's associated storage area.
- The virtual machine is registered with the new host.
- After the migration is completed, the old version of the virtual machine is deleted from the source host

BITS Pilani

Storage Migration

- This kind of migration constitutes moving the virtual disks or configuration file of a running virtual machine to a new data store without any interruption in the availability of the virtual machine's service
- 1. Throughout most of the move operation, disk reads and writes go to the source virtual hard disk.
- 2. While reads and writes occur on the source virtual hard disk, the disk contents are copied to the new destination virtual hard disk.
- 3. After the initial disk copy is complete, disk writes are mirrored to both the source and destination virtual hard disks while outstanding disk changes are replicated.
- 4. After the source and destination virtual hard disks are completely synchronized, the virtual machine switches over to using the destination virtual hard disk.
- 5 The source virtual hard disk is deleted.



FUTURE DIRECTIONS

- Self-adaptive and dynamic data center.
- Performance evaluation and workload characterization of virtual workloads.
- High-performance data scaling in private and public cloud environments.
- Performance and high availability in clustered VMs through live migration.
 - VM scheduling algorithms.
 - Accelerating VMs live migration time.
 - Cloud-wide VM migration and memory de-duplication.
- Live migration security.

BITS Pilani

Agenda

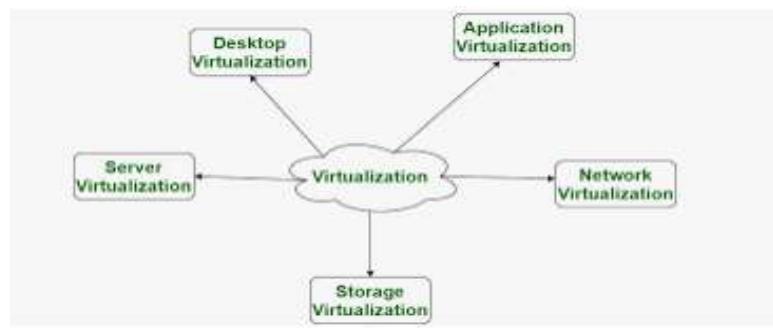
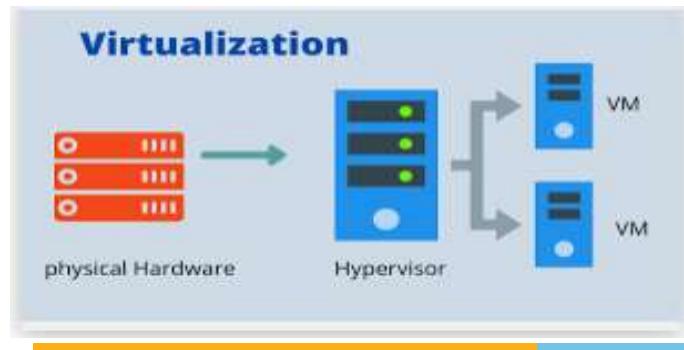
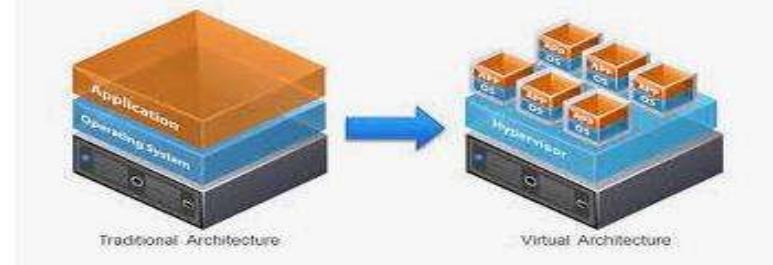


- ❖ Virtualization Recap
- ❖ Containers
 - ❖ Introduction & Motivation for Containers
 - ❖ Linux Container –LXC and LXD
 - ❖ Container Architecture
 - ❖ Orchestration technologies
 - ❖ Docker Container and Components
 - ❖ Hands on with Docker

Virtualization

What Is Virtualization ?

- Virtualization is a Technology that transforms hardware into software.
- Virtualization allows to run multiple operating systems as virtual machines.
- Each copy of an operating system is installed in to a virtual machine.



BITS Pilani

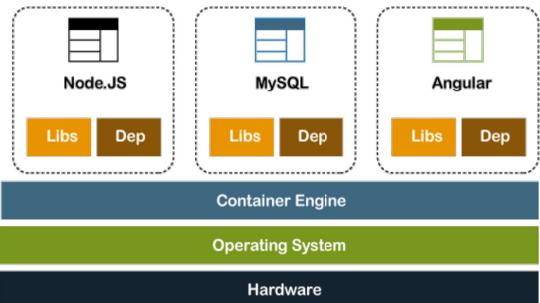


BITS Pilani

Pilani | Dubai | Goa | Hyderabad



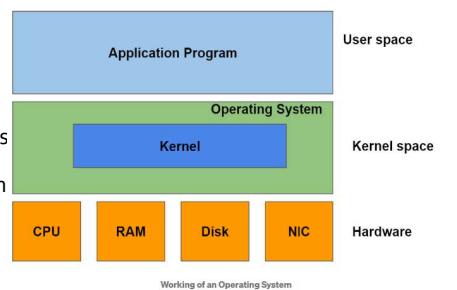
Containers





Motivation towards Containers

- Let's assume that you are building a web service on an Ubuntu machine. Your code works fine on your local machine. You have a remote server in your data centre that can run your application.
- You copy your local binaries on the remote server and try to run your code. The next thing that you see is your code doesn't work there. The above problems result in portability issues. The developer has to spend a lot of time debugging the environment-specific issues.



- The computer has different hardware resources such as RAM, hard disk, Network Interface Card, IO devices, etc. The operating system is the software that manages this hardware.
- OS consists of a system program known as the kernel, which is loaded in the memory when OS starts. The kernel is responsible for process management, CPU scheduling, file system & IO.
- User programs interact with the hardware through the means of the kernel. For eg:- Let's say your application wants to open a file and write content in it. The application will invoke system calls like fopen() and fwrite() to perform its functions.
- The kernel performs the function on behalf of the user program and gives the output back to it. The following diagram shows the different layers involved in the functioning of an application program.

BITS Pilani

What are Containers?

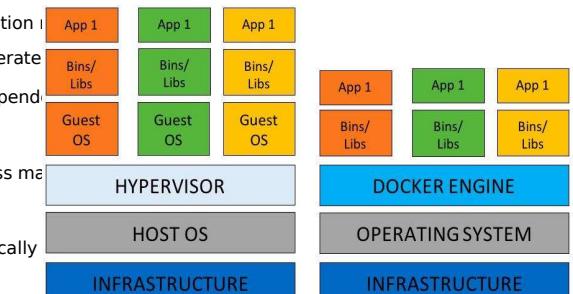
A container is a standard unit of software that packages up code and all its dependencies so the application runs reliably from one computing environment to another. The way which containerized applications operate is called containerization. Containers are an operating system virtualization technology used to package applications and their dependencies in isolated environments.

They provide a lightweight method of packaging and deploying applications in a standardized way across many types of infrastructure.

Containers run consistently on any container-capable host, so developers can test the same software locally and deploy to full production environments.

The container format also ensures that the application dependencies are baked into the image itself, simplifying the hand off and release processes.

Because the hosts and platforms that run containers are generic, infrastructure management for container-based systems can be standardized.



As can be seen from this diagram, a container includes an application plus any binaries or libraries that the application requires in order for it to run.

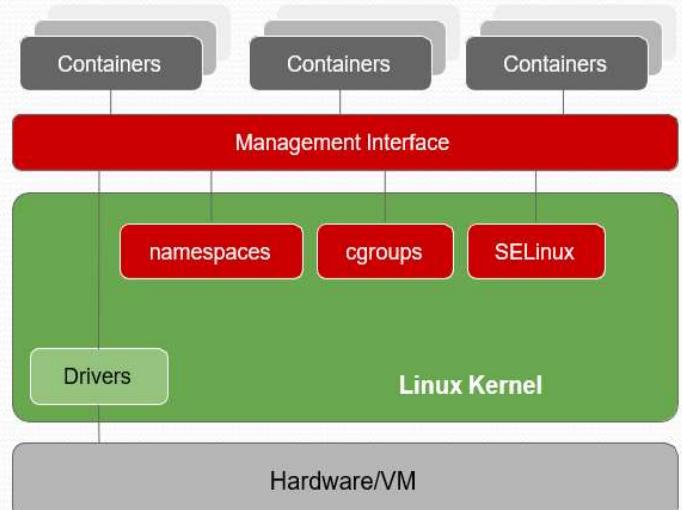
The container runs under the control of the container engine (such as Docker or CRI-O), which in turn runs on top of the operating system (which can be Windows 10, Windows Server 2016, or Linux depending on the container engine being used).

Linux Containers

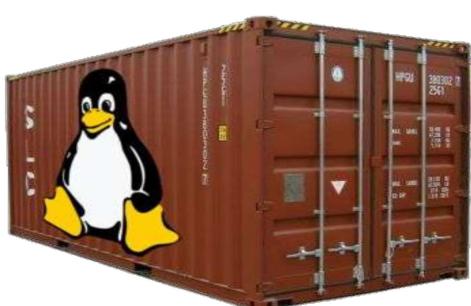
 Linux Containers(LXC) allow running multiple isolated Linux instances (containers) on the same host.

 Containers share the same kernel with anything else that is running on it, but can be constrained to only use a defined amount of resources such as CPU, memory or I/O.

 A container is a way to isolate a group of processes from the others on a running Linux system.



Linux Containers LXC

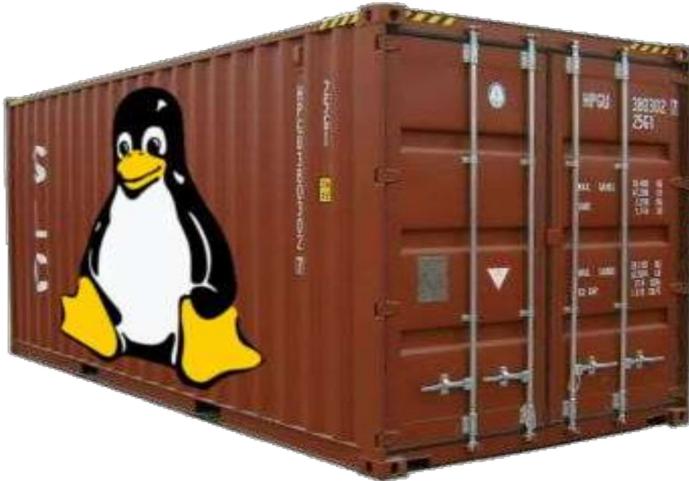


- LXC is an abbreviation used for Linux Containers which is an operating system that is used for running multiple Linux systems virtually on a controlled host via a single Linux kernel.
- LXC bundles with the kernel's Cgroups to provide the functionality for the process and network space instead of creating a full virtual machine and provides an isolated environment for the applications.

- Features provided by LXC :

- It provides Kernel namespaces such as IPC, mount, PID, network, and user.
- It provides Kernel capabilities.
- Control groups (Cgroups).
- Seccomp profiles

Linux Containers LXC



- Lightweight virtualization.
- OS-level virtualization
- Allow single host to operate multiple **isolated & resource-controlled** Linux Instances.
- included in the Linux kernel called LXC (Linux Container)

Containers are not a new technology: the earliest iterations of containers have been around in open source Linux code for decades.

BITS Pilani

Control Groups

Control Groups or **cgroups** was introduced by Google in 2006 which was based on Linux Kernel Feature.

By using cgroups, system administrators gain fine-grained control over allocating, prioritizing, denying, managing, and monitoring system resources.

All processes on a Linux system are child processes of a common parent: the **init process**, which is executed by the kernel at boot time and starts other processes (which may in turn start child processes of their own). Because all processes descend from a single parent, the Linux process model is a single hierarchy, or tree.

Cgroups are similar to processes in that:

- they are hierarchical, and
- child cgroups inherit certain attributes from their parent cgroup.

The fundamental difference is that many different hierarchies of cgroups can exist simultaneously on a system. If the Linux process model is a single tree of processes, then the cgroup model is one or more separate, unconnected trees of tasks (i.e. processes).

Multiple separate hierarchies of cgroups are necessary because each hierarchy is attached to **one or more subsystems**. A subsystem represents a single resource, such as CPU time or memory. Red Hat Enterprise Linux 6 provides ten cgroup subsystems, listed below by name and function.

Control Groups

- `blkio` – this subsystem sets limits on input/output access to and from block devices such as physical drives (disk, solid state, or USB).
- `cpu` – this subsystem uses the scheduler to provide cgroup tasks access to the CPU.
- `cputat` – this subsystem generates automatic reports on CPU resources used by tasks in a cgroup.
- `cpuset` – this subsystem assigns individual CPUs (on a multicore system) and memory nodes to tasks in a cgroup.
- `devices` – this subsystem allows or denies access to devices by tasks in a cgroup.
- `freeser` – this subsystem suspends or resumes tasks in a cgroup.
- `memory` – this subsystem sets limits on memory use by tasks in a cgroup and generates automatic reports on memory resources used by those tasks.
- `net_cls` – this subsystem tags network packets with a class identifier (classid) that allows the Linux traffic controller (`tc`) to identify packets originating from a particular cgroup task.
- `net_prio` – this subsystem provides a way to dynamically set the priority of network traffic per network interface.
- `ns` – the namespace subsystem.
- `perf_event` – this subsystem identifies cgroup membership of tasks and can be used for performance analysis.

cgroup: Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour.

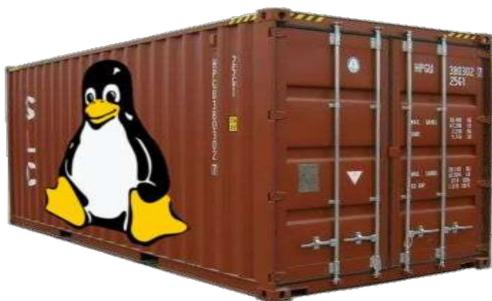
namespace: wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource.

In short:

- Cgroups = limits how much you can use;
- namespaces = limits what you can see (and therefore use)

BITS Pilani

Linux Containers LXD



- The simplest way to define LXD is to say it's an extension of LXC. LXD also happens to be LXC's main claim to fame, now that LXC has ceased to be important for Docker and CoreOS.
- The more technical way to define LXD is to describe it as a REST API that connects to libxl, the LXC software library. LXD, which is written in Go, creates a system daemon that apps can access locally using a Unix socket, or over the network via HTTPS.

A host can run many LXC containers using only a single system daemon, which simplifies management and reduces overhead. With pure-play LXC, you'd need separate processes for each container.

The LXD daemon can take advantage of host-level security features to make containers more secure. On plain LXC, container security is more problematic.

Because the LXD daemon handles networking and data storage, and users can control these things from the LXD CLI interface, it simplifies the process of sharing these resources with containers.

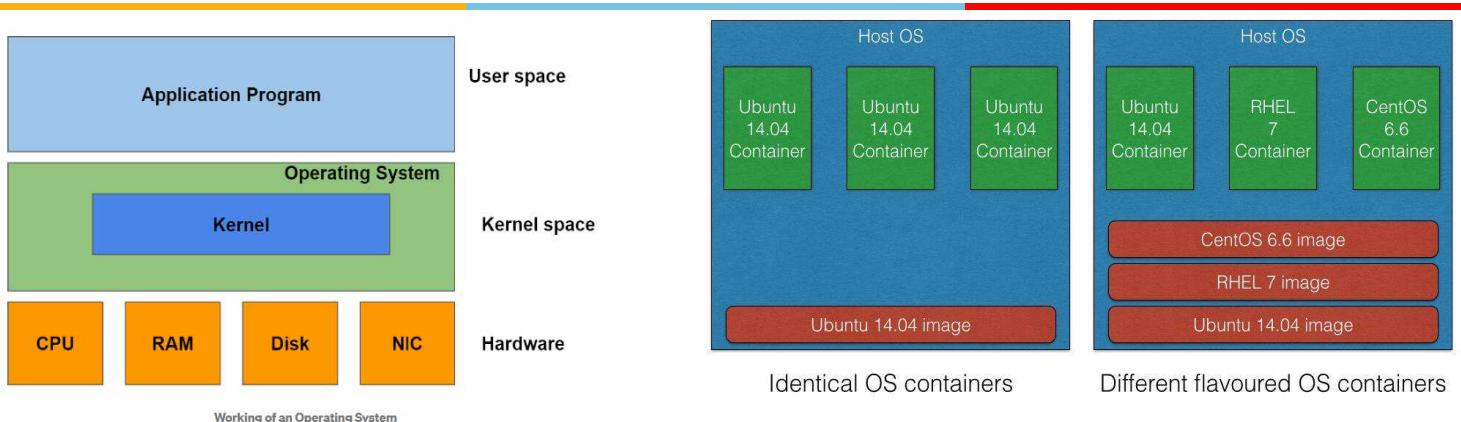
LXD offers advanced features not available from LXC, including live container migration and the ability to snapshot a running container.

Comparing LXC & LXD

LXC is a virtual environment creation tool, it was built by Google, IBM etc.	LXD is an add on for the LXC to provide advanced features and functionalities.
Multiple processes are needed for multiple containers and hence it is not flexible.	LXD makes it flexible by providing a single process for multiple containers.
Snapshots, Live Migration etc are some of the features which are not supported by LXC.	LXD supports snapshots and lives migration features.
Scalability functionality is not provided by LXC and hence users shift to other virtual solutions.	With the use of LXD, scalability is achieved in LXC.
Management capabilities are poor, especially in the case of it has better management capabilities like storage pooling, network and storage.	
It is not user friendly and needs the expertise to handle thet provides a user-friendly interface.	
After data processing, the data cannot be retrieved.	Data retrieval functionality after data processing is provided in LXD.
C API is used by the LXC.	LXD uses REST API.

BITS Pilani

Container Types – OS Containers



- OS containers are virtual environments that share the kernel of the host operating system but provide user space isolation.
- For all practical purposes, you can think of OS containers as VMs. You can install, configure and run different applications, libraries, etc., just as you would on any OS.
- Just as a VM, anything running inside a container can only see resources that have been assigned to that container.
- OS containers are useful when you want to run a fleet of identical or different flavors of distros.
- Most of the times containers are created from templates or images that determine the structure and contents of the container.
- It thus allows you to create containers that have identical environments with the same package versions and configurations across all containers.

Minimalistic OS

A common set of ideas:

A minimal operating system is one that has been stripped of all unnecessary components and provides only the functionality needed for a specific purpose.

- Stability is enhanced through transactional upgrade/rollback semantics.
- Traditional package managers are absent and may be replaced by new packaging systems (Snappy), or custom image builds (Atomic).
- Security is enhanced through various isolation mechanisms

	CoreOS (647.0.0)	RancherOS (0.23.0)	Atomic (F 22)	Photon	Snappy (edge – 145)
Size	164MB	20MB	151/333MB	251MB	111MB
Kernel version	3.19.3	3.19.2	4.0.0	3.19.2	3.18.0
Docker version	1.5.0	1.6.0	1.6.0	1.5.0	1.5.0
Init system	systemd	Docker	systemd	systemd	systemd
Package manager	None (Docker/Rocket)	None (Docker)	Atomic	dnf (tyum)	Snappy
Filesystem	ext4	ext4	xfs	ext4	ext4
Tools	Fleet, etcd	–	Cockpit (Anaconda, kickstart), atomic	–	–

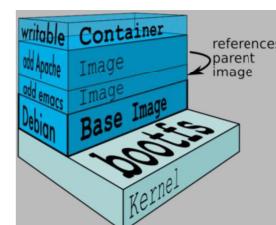
BITS Pilani

Container Types – Application Containers

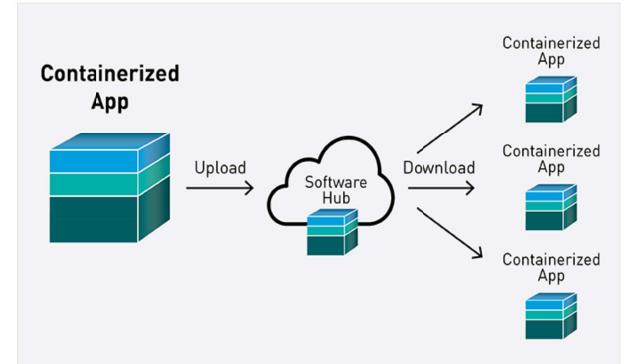
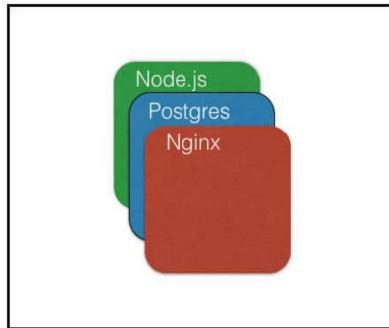
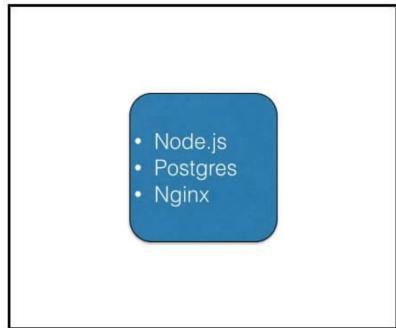
APPLICATION CONTAINERS

- Designed to package and run a single service
- All containers share host kernel
- Subtle differences from operating system containers
- Examples: Docker, Rocket
- Docker: runs a single process on creation
- OS containers: run many OS services, for an entire OS
- Create application containers for each component of an app
- Supports a micro-services architecture
- DevOPS: developers can package their own components in application containers
- Supports horizontal and vertical scaling

- While OS containers are designed to run multiple processes and services, application containers are designed to package and run a single service.
- Container technologies like Docker and Rocket are examples of application containers.
- So even though they share the same kernel of the host there are subtle differences make them different, which I would like to talk about using the example of a Docker container:



Comparison OS vs Application Containers

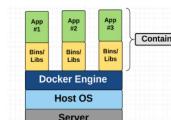
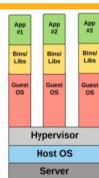


- Meant to be used as an OS - run multiple services
- No layered filesystems by default
- Built on cgroups, namespaces, native process resource isolation
- Examples - LXC, OpenVZ, Linux VServer, BSD Jails, Solaris Zones

- Meant to run for a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples - Docker, Rocket

BITS Pilani

Difference between VM & Container



VM

Virtual machines, or VMs, are a **hardware virtualization technology** that allows you to fully virtualize the **hardware and resources of a computer**.

A **separate guest operating system** manages the **virtual machine**, separate from the OS running on the host system.

On the **host system**, a piece of software called a **hypervisor** is responsible for starting, stopping, and **managing** the virtual machines.

Because VMs are operated as completely distinct computers that, under normal operating conditions, cannot affect the host system or other VMs, virtual machines offer great isolation and security.

In general, virtual machines let you subdivide a machine's resources into smaller, individual computers, but the result doesn't differ significantly from managing a fleet of physical computers.

Container

Containers take a different approach. Rather than virtualizing the entire computer, **containers virtualize the operating system directly**.

They run as **specialized processes managed by the host operating system's kernel**, but with a **constrained and heavily manipulated view** of the system's processes, resources, and environment.

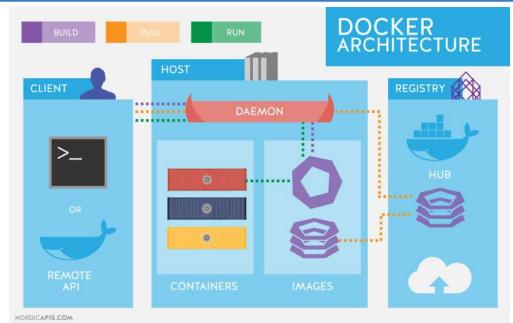
Containers are **unaware** that they exist on a **shared system** and **operate** as if they were in **full control of the computer**.

it is more common to manage containers more similarly to applications.

Containers occupy a space that sits somewhere in between the strong isolation of virtual machines and the native management of conventional processes.



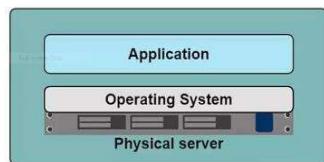
Dockers



Dockers - Motivation

Problems in the Past

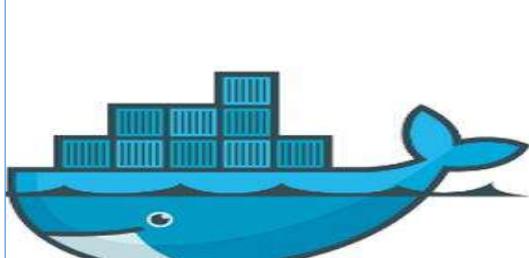
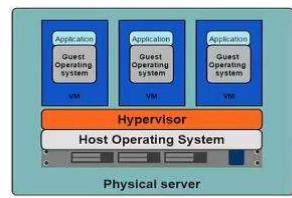
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



Virtualization →

Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



Containers ←

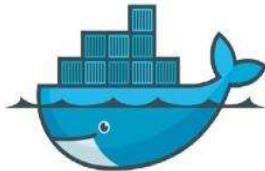
Limitations of VMs

- Each VM stills requires
 - CPU allocation
 - Storage
 - RAM
 - An entire guest operating system
- The more VM's you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed



Dockers - Introduction

What Is Docker?



- Lightweight, open, secure platform
- Simplify building, shipping, running apps
- Runs natively on Linux or Windows Server
- Runs on Windows or Mac Development machines (with a virtual machine)
- Relies on "images" and "containers"



- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. At a high level, Docker is a utility that can efficiently create, ship, and run containers.
- Docker containers wrap a piece of software in a complete file system that contains everything needed to run: code, runtime, system tools, system libraries
- Docker enables you to quickly, reliably, and consistently deploy applications regardless of environment.
- Docker enables you to **achieve compute density** by running several **isolated containers on the same hardware**.
- Namespaces provides the isolation

BITS Pilani

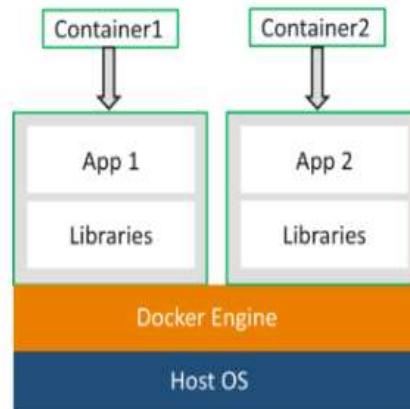
Dockers - Introduction

What is Docker?

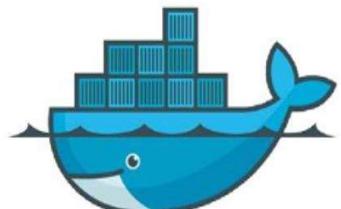
Docker is a platform which packages an application and all its dependencies together in the form of containers. This containerization aspect ensures that the application works in any environment.

As you can see in the diagram, each and every application runs on separate containers and has its own set of dependencies & libraries. This makes sure that each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

So a developer can build a container having different applications installed on it and give it to the QA team. Then the QA team would only need to run the container to replicate the developer's environment.



What Is Docker?



Dockers - Architecture

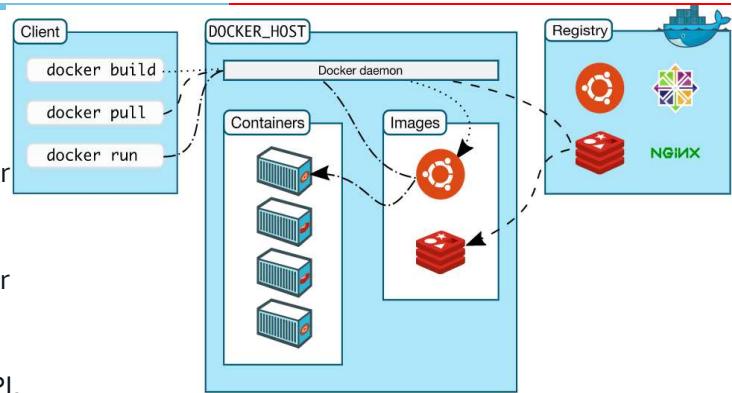
Docker uses a client-server architecture.

The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.

The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon.

The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



BITS Pilani

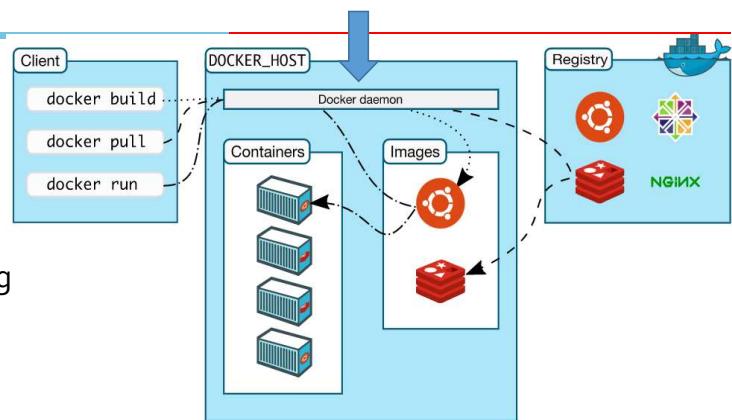
Dockers – Components - Daemon

The Docker daemon, also known as 'dockerd', consistently listens to the requests put forward by the Docker API.

It is used to carry out all the heavy tasks such as creating and managing Docker objects including containers, volumes, images, and networks.

A Docker daemon is also capable of communicating with other daemons in the same or different host machines.

For example, in the case of a swarm cluster, the host machine's daemon can communicate with daemons on other nodes to carry out tasks.

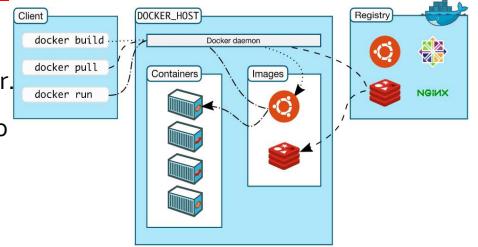


Dockers – Components – Client

The **Docker client** (`docker`) is the primary way that many Docker users interact with Docker.

The Docker users can leverage simple HTTP clients like Command line to interact with Docker.

When a user executes a Docker command such as Docker run, the CLI will send this request to the dockerd via the REST API. The Docker CLI can also communicate with over one daemon.



Docker Desktop

Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices. Docker Desktop includes the Docker daemon (`dockerd`), the Docker client (`docker`), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper. For more information, see [Docker Desktop](#).

Docker registries

A Docker *registry* stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the `docker pull` or `docker run` commands, the required images are pulled from your configured registry. When you use the `docker push` command, your image is pushed to your configured registry.

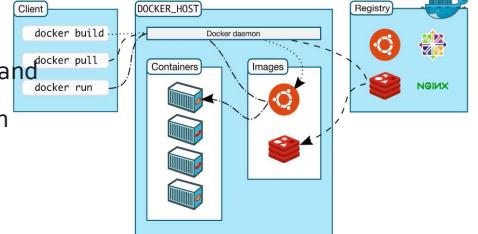
BITS Pilani

Dockers – Components – Registries

Docker registries

A Docker *registry* stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the `docker pull` or `docker run` commands, the required images are pulled from your configured registry. When you use the `docker push` command, your image is pushed to your configured registry.



The official [Docker registry](#) called Dockerhub contains several official image repositories.

A repository contains a set of similar Docker images that are uniquely identified by Docker tags.

Dockerhub provides tons of useful official and vendor-specific images to its users.

Some of them include Nginx, Apache, Python, Java, Mongo, Node, MySQL, Ubuntu, Fedora, Centos, etc.

You can even create your private repository inside Dockerhub and store your custom Docker images using the Docker push command.

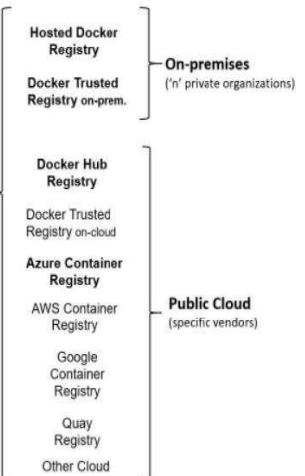
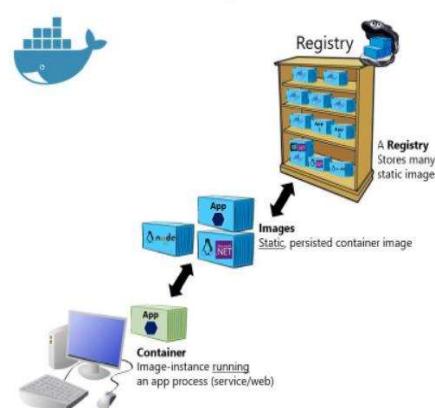
Docker Objects

A Docker user frequently interacts with

Docker objects such as

- Images
- Containers
- Volumes
- Plugins
- Networks and so on.

Basic taxonomy in Docker



BITS Pilani

Docker Objects - Images

Docker Images are read-only templates that are built using multi-layers of file.

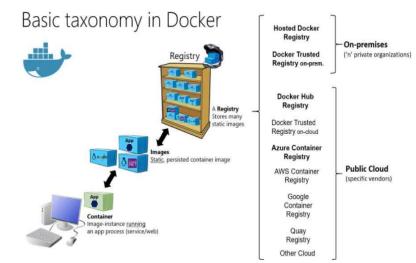
You can build Docker images using a simple text file called Dockerfile which contains instructions to build Docker images.

The first instruction is a FROM instruction which can pull a base image from any Docker registry. Once this base image layer is created, several instructions are then used to create the container environment. Each instruction adds a new layer on top of the previous one.

A Docker image is simply a **blueprint of the container environment**. Once you create a container, it creates a writable layer on top of the image, and then, you can make changes.

The images all the metadata that describes the container environment. You can either directly pull a Docker image from Dockerhub or create your customized image over a base image using a Dockerfile.

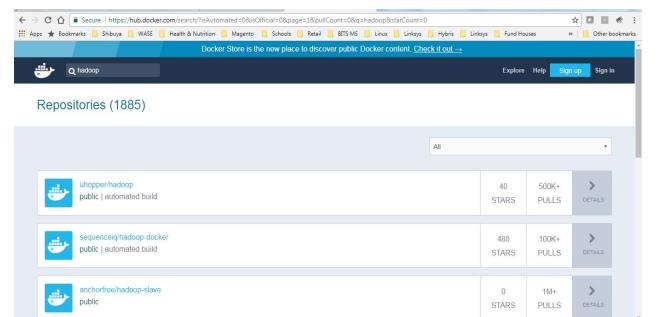
Once you have created a Docker image, you can push it on **Dockerhub** or any other registry and share it with the outside world.



Docker Objects - Images

docker image index: a repository (public or private) for docker images

- Docker images constitute the base of docker containers *from which everything starts to form*. They are very similar to default operating-system disk images which are used to run applications on servers or desktop computers.
- Having these images (e.g. Ubuntu base) allow seamless portability across systems. They make a solid, consistent and dependable base with everything that is needed to run the applications. When everything is self-contained and the risk of system-level updates or modifications are eliminated, the container becomes immune to external exposures which could put it out of order - *preventing the dependency hell*.



Note: Repositories contain images from un verified sources. Exercise caution. Always use the official images

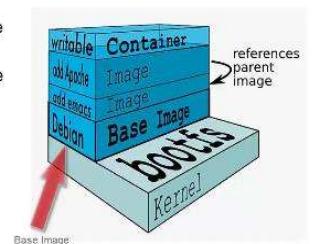
BITS Pilani

Docker Objects - Images

- As more layers (tools, applications etc.) are added on top of the base, new images can be formed by *committing* these changes. When a new container gets created from a saved (i.e. committed) image, things continue from where they left off. And the [union file system](#), brings all the layers together as a single entity when you work with a container.
- These base images can be explicitly stated when working with the **docker CLI** to directly create a new container or they might be specified inside a **Dockerfile** for automated image building.

Image Layers

- Images are comprised of multiple layers
- A layer is also just another image
- Every image contains a base layer
- Docker uses a copy on write system
- Layers are read only

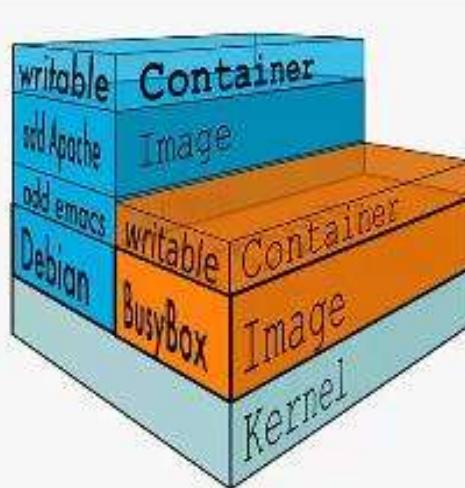


Note: Repositories contain images from un verified sources. Exercise caution. Always use the official images

Docker Objects - Images

The Container Writable Layer

- Docker creates a top writable layer for containers
- Parent images are read only
- All changes are made at the writable layer



BITS Pilani

Docker Objects - Containers

Docker containers are isolated, encapsulated, packaged, and secured application environments that contain all the packages, libraries, and dependencies required to run an application.

For example, if you create a container associated with the Ubuntu image, you will have access to an isolated Ubuntu environment. You can also access the bash of this Ubuntu environment and execute commands.

Containers have all the access to the resources that you define while using the **Dockerfile** while creating an image. Such configurations include *build context*, *network connections*, *storage*, *CPU*, *memory*, *ports*, etc.

For example, if you want access to a container with libraries of Java installed, you can use the Java image from the Dockerhub and run a container associated with this image using the Docker run command.

You can also create containers associated with the custom images that you create for your application using the Dockerfiles. Containers are very light and can be spun within a matter of seconds.

Creating a Container

- Use `docker run` command
- Syntax
`sudo docker run [options] [image] [command] [args]`
- Image is specified with `repository:tag`

Examples

```
docker run ubuntu:14.04 echo "Hello World"  
docker run ubuntu ps ax
```

Docker Objects - DockerFiles

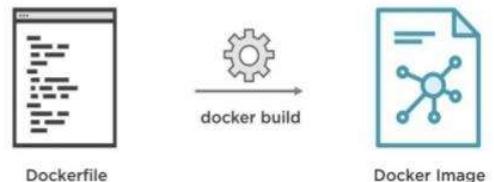
An alternative to the previous image build method using docker commit

Dockerfiles are scripts containing a successive series of instructions, directions, and commands which are to be executed to form a new docker image.

Each command executed translates to a new layer of the onion, forming the end product.

They basically replace the process of doing everything manually and repeatedly.

When a **Dockerfile** is finished executing, you end up having formed an image, which then you use to start (i.e. create) a new container.



BITS Pilani

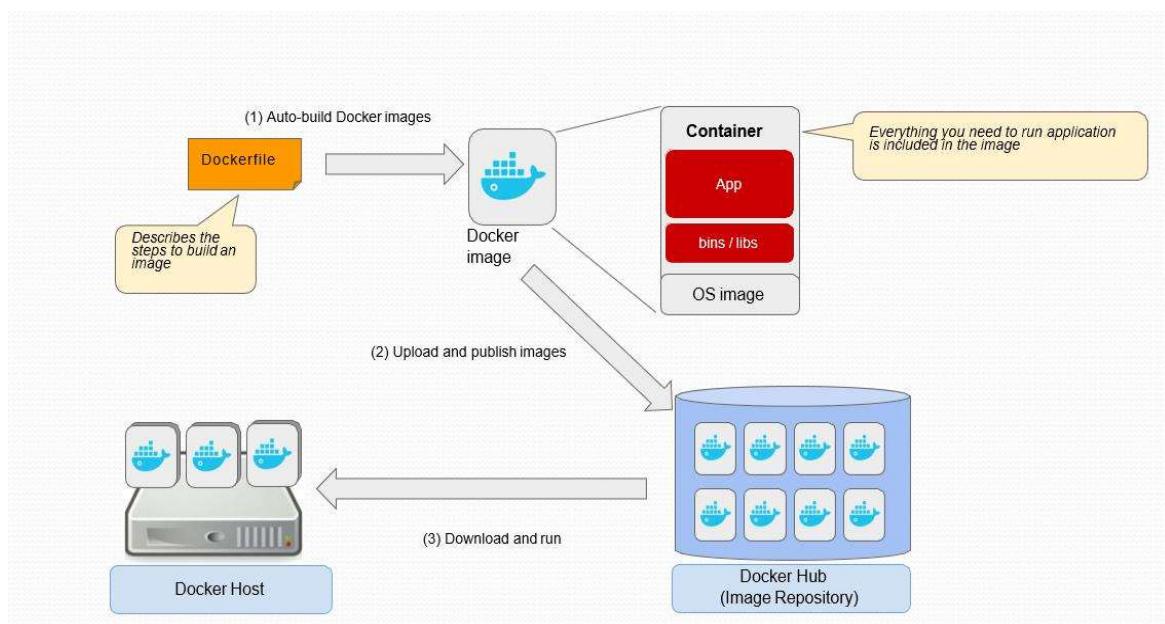
Docker Objects - DockerFiles

Dockerfile Instructions

- Instructions specify what to do when building the image
- **FROM** instruction specifies what the base image should be
- **RUN** instruction specifies a command to execute

```
#Example of a comment
FROM ubuntu:14.04
RUN apt-get install vim
RUN apt-get install curl
```

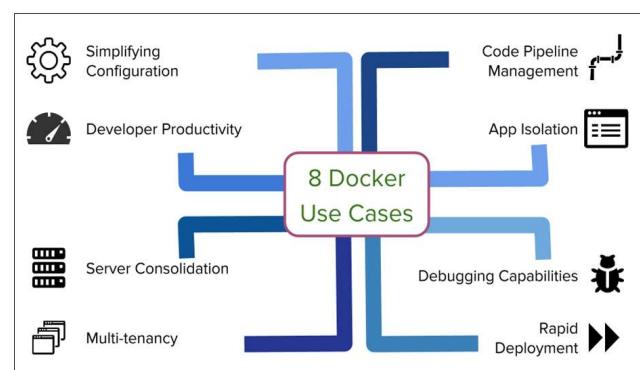
Docker Life Cycle



BITS Pilani

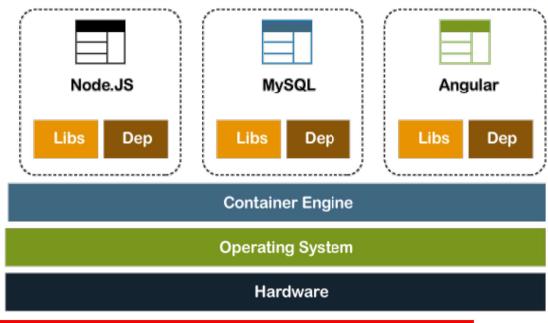
Docker Summary

- A **container** is one or more runtime instances of a Docker image that usually will contain a single app/service. The container is considered the live artifact being executed in a development machine or the cloud or server.
- An **image** is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a union of layered filesystems (deltas) stacked on top of each other. An image does not have state and it never changes.
- A **registry** is a service containing repositories of images from one or more development teams. Multiple development teams may also instance multiple registries. The default registry for Docker is the public "Docker Hub" but you will likely have your own private registry network close to your orchestrator to manage and secure your images, and reduce network latency when deploying images.
- The beauty of the images and the registry resides on the possibility for you to store static and immutable application bits including all their dependencies at OS and frameworks level so they can be versioned and deployed in multiple environments providing a consistent deployment unit.





Container Orchestration

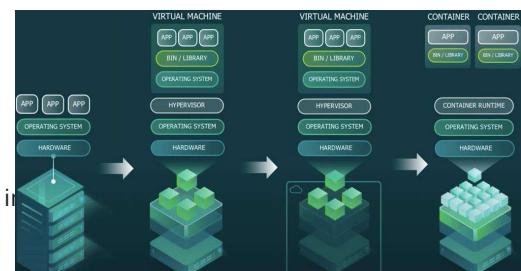


Container Orchestration

Container orchestration is all about managing the lifecycles of containers, especially in large, dynamic environments.

Software teams use container orchestration to control and automate many tasks:

- Provisioning and deployment of containers
- Scaling up or removing containers to spread application load evenly across host infrastructure
- Movement of containers from one host to another if there is a shortage of resources in host, or if a host dies
- External exposure of services running in a container with the outside world
- Load balancing of service discovery between containers
- Health monitoring of containers and hosts
- Configuration of an application in relation to the containers running it



Container Orchestration

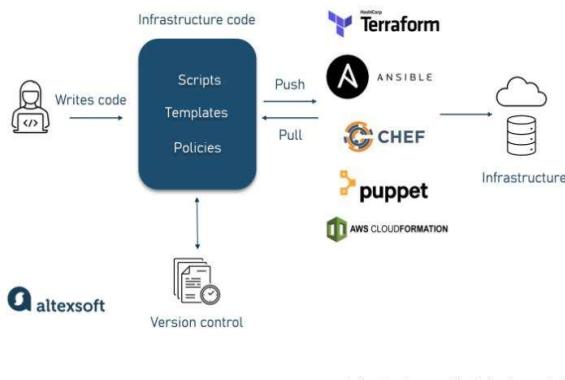
KEY ORCHESTRATION FEATURES

- Management of container hosts
- Launching set of containers
- Rescheduling failed containers
- Linking containers to support workflows
- Providing connectivity to clients outside the container cluster
- Firewall: control network/port accessibility
- Dynamic scaling of containers: horizontal scaling
 - Scale in/out, add/remove containers
- Load balancing over groups of containers
- Rolling upgrades of containers for application

BITS Pilani

IAC

HOW INFRASTRUCTURE AS CODE WORKS



COMPARING INFRASTRUCTURE AS CODE TOOLS

Name	Languages	Function	Approach	Infrastructure type
Terraform	HCL + Typescript, Python, Java, C#, Go with CDK	Provisioning	Declarative	Immutable
AWS CloudFormation	JSON, YAML + TypeScript, Python, Java, .NET, and Go with CDK	Provisioning	Declarative	Both
Ansible	Python, Ruby, YAML	Configuration management	Imperative	Mutable
Puppet	PuppetDSL, YAML	Configuration management	Declarative	Mutable
Chef	Ruby	Configuration management	Declarative	Mutable

- 1.The team writes infrastructure configurations in a required programming language.
- 2.The code files are sent to a code repository.
- 3.An IaC tool runs the code and performs the required activities.

Imperative vs Declarative Model in IAC

The terms *imperative* and *declarative* come up frequently in IAC discussions.

Both terms refer to how the user provides direction to the automation platform.

With an imperative tool, you define the steps to execute in order to reach the desired solution.

With a declarative tool, you define the desired state of the final solution, and the automation platform determines how to achieve that state.

Imperative systems are often initially easier.

You could say that an imperative system is organized more like how a human thinks.

Imperative systems allow you to continue to think about configuration as a series of actions or steps, each bringing you closer to the goal.

Another benefit of imperative language is that it allows you to automate very detailed and complex configurations by building up multiple layers of commands. And, because an imperative system gives the user more control over how to accomplish a task, it is often more efficient and easier to optimize for a specific purpose than a declarative system.

BITS Pilani

Imperative vs Declarative Model in IAC

With all the advantages of imperative languages, you might be wondering why declarative languages are so much in vogue.

Declarative tools have been gaining popularity for several years now and arguably are the dominant format for IaC automation.

One reason for the popularity of declarative tools is that they require less knowledge on the user's part, at least once you understand how they work.

With an imperative tool, the user must have enough knowledge to tell the automation platform what to do.

With a declarative system, the user only needs to define the state of the final configuration, and the platform determines how to get there. The complexity of the step-by-step execution is hidden from the user.

Another benefit of a declarative language is that it is more *idempotent*. The concept of idempotence refers to a process that can be executed multiple times with the same result. Because a declarative language just defines the final state, you always end up in the same place no matter where you start. On the other hand, an imperative language envisions a task as a series of predefined steps that could take you to a different endpoint depending on the starting point.

Imperative vs Declarative Model in IAC

Think about **imperative configuration** like a remote control for In our analogy, the **declarative model** is like a thermostat. a television.

You use the buttons on it to change things until it's how you want it to be. Make the sound louder, change the channel, turn on recording, etc.

Simply put, **imperative** is the "**how**" of configuration paradigms.

In an edge API Gateway, you can go to an interface or a REST API to specifically enter the timeout for routing rules.

First, it's 8 seconds. Then later on, you change it to 5 seconds.

You embed how you want it to work into the control flow of your API or the program calling it.

You know that you'll be cold until it's 70F so you set the temperature to 70F and you go about your day. The thermostat detects its environment and controls the temperature accordingly.

Declarative is **expressing intent** of "**how**" you want your **system to work**.

This model uses a GitOps style approach, because you can write down in your source file that the timeout is 10 seconds and that is committed to source control. Your continuous integration system reads that source file and deploys it to your Kubernetes cluster. So, Git becomes your source of truth.

BITS Pilani

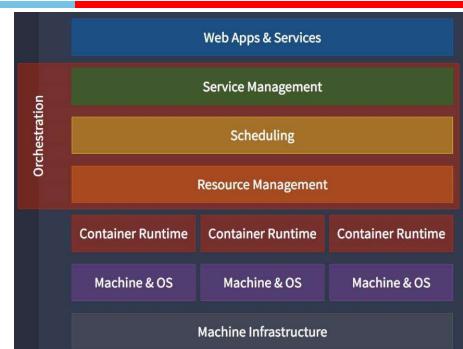
Container Orchestration

Most container orchestration tools support a **declarative configuration model**:

A developer writes a **configuration file** (in YAML or JSON depending on the tool) that defines a desired configuration state, and the orchestration tool runs the file uses its own intelligence to achieve that state. The configuration file typically

- Defines which container images make up the application, and where they are located (in what registry)
- Provisions the containers with storage and other resources
- Defines and secures the network connections between containers
- Specifies versioning (for phased or canary rollouts)

The orchestration tool schedules deployment of the containers (and replicas of the containers, for resiliency) to a host, choosing the best host based on available CPU capacity, memory, or other requirements or constraints specified in the configuration file.



Container orchestration mediates between the apps or services above and the container runtimes below. Three main functional aspects of what they do include:*

- **Service Management:** Labels, groups, namespaces, dependencies, load balancing, readiness checks.
- **Scheduling:** Allocation, replication, resurrection, rescheduling, rolling deployment, upgrades, downgrades.
- **Resource Management:** Memory, CPU, GPU, volumes, ports, IPs.

Kubernetes – Introduction Video



Kubernetes a self-service Platform-as-a-Service (PaaS) that creates a hardware layer abstraction for development teams. Kubernetes is also extremely portable. It runs on [Amazon Web Services \(AWS\)](#), [Microsoft Azure](#), [the Google Cloud Platform \(GCP\)](#), or in on-premise installations. we can move workloads without having to redesign your applications or completely rethink your infrastructure—which helps you to standardize on a platform and avoid vendor lock-in.

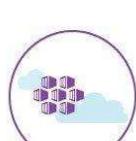
BITS Pilani

Container Orchestration - Tools

Tools of Container Orchestration



Amazon ECS
FROM AMAZON



Azure Container Services
FROM MICROSOFT



Docker Swarm
DOCKER OPENSOURCE TOOLS



Google Container Engine
FROM GOOGLE CLOUD PLATFORM



Kubernetes
DOCKER OPENSOURCE TOOLS



CoreOS Fleet
FROM COREOS



Mesosphere Marathon
FROM MARATHON



Cloud Foundry's Diego
FROM CLOUD FOUNDRY

- **Docker Swarm:** Provides native clustering functionality for Docker containers, which turns a group of Docker engines into a single, virtual Docker engine.
- **Google Container Engine:** Google Container Engine, built on Kubernetes, lets you run Docker containers on the Google Cloud.
- **Kubernetes:** An orchestration system for Docker containers. It handles scheduling and manages workloads based on user-defined parameters.
- **Mesosphere Marathon:** Marathon is a container orchestration framework for Apache Mesos that is designed to launch long-running applications.
- **Amazon ECS:** The ECS supports Docker containers and lets you run applications on a managed cluster of Amazon EC2 instances.
- **Azure Container Service (ACS):** ACS lets you create a cluster of virtual machines that act as container hosts along with master machines that are used to manage your application containers.
- **Cloud Foundry's Diego:** Container management system that combines a scheduler, runner, and health manager.
- **CoreOS Fleet:** Container management tool that lets you deploy Docker containers on hosts in a cluster as well as distribute services across a cluster.

Agenda



- ❖ Container Recap
- ❖ Introduction to PaaS
- ❖ Building blocks of PaaS
- ❖ Characteristics of PaaS
- ❖ Advantages and Risks
- ❖ PaaS Example – Windows Azure

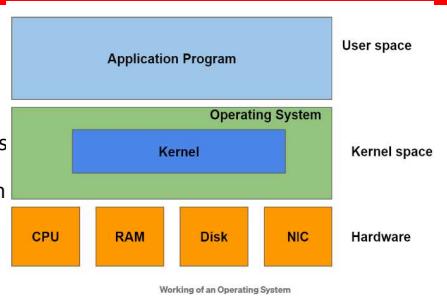
2

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

Motivation towards Containers



- Let's assume that you are building a web service on an Ubuntu machine. Your code works fine on your local machine. You have a remote server in your data centre that can run your application.
- You copy your local binaries on the remote server and try to run your code. The next thing that you see is your code doesn't work there. The above problems result in portability issues. The developer has to spend a lot of time debugging the environment-specific issues.
- The computer has different hardware resources such as RAM, hard disk, Network Interface Card, IO devices, etc. The operating system is the software that manages this hardware.
- OS consists of a system program known as the kernel, which is loaded in the memory when OS starts. The kernel is responsible for process management, CPU scheduling, file system & IO.
- User programs interact with the hardware through the means of the kernel. For eg:- Let's say your application wants to open a file and write content in it. The application will invoke system calls like fopen() and fwrite() to perform its functions.
- The kernel performs the function on behalf of the user program and gives the output back to it. The following diagram shows the different layers involved in the functioning of an application program.



What are Containers?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. The way which containerized applications operate is shown→

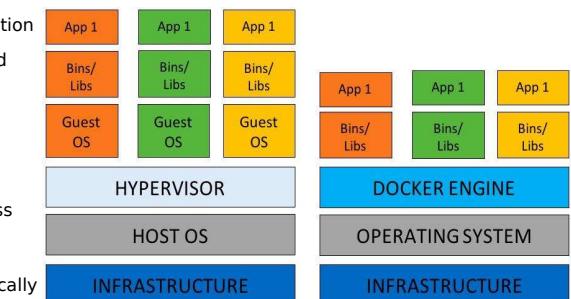
Containers are an operating system virtualization technology used to package applications and their dependencies and run them in isolated environments.

They provide a lightweight method of packaging and deploying applications in a standardized way across many different types of infrastructure

Containers run consistently on any container-capable host, so developers can test the same software locally that they will later deploy to full production environments.

The container format also ensures that the application dependencies are baked into the image itself, simplifying the hand off and release processes.

Because the hosts and platforms that run containers are generic, infrastructure management for container-based systems can be standardized.



As can be seen from this diagram, a container includes an application plus any binaries or libraries that the application requires in order for it to run.

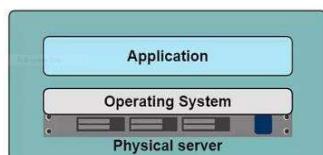
The container runs under the control of the container engine (such as Docker or CRI-O), which in turn runs on top of the operating system (which can be Windows 10, Windows Server 2016, or Linux depending on the container engine being used).

BITS Pilani

Dockers - Motivation

Problems in the Past

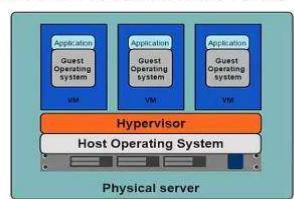
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



Virtualization →

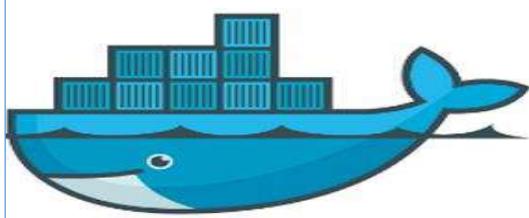
Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



Limitations of VMs

- Each VM stills requires
 - CPU allocation
 - Storage
 - RAM
 - An entire guest operating system
- The more VM's you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed



Containers ←

Difference between VM & Container



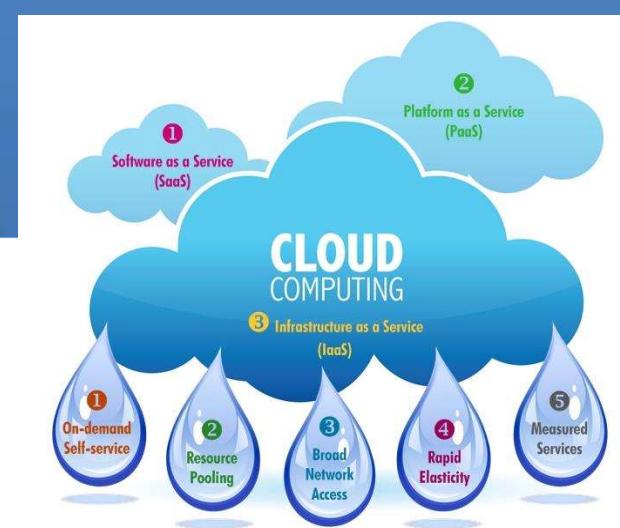
VM	Container
Virtual machines, or VMs, are a hardware virtualization technology that allows you to fully virtualize the hardware and resources of a computer .	Containers take a different approach. Rather than virtualizing the entire computer, containers virtualize the operating system directly.
A separate guest operating system manages the virtual machine , separate from the OS running on the host system.	They run as specialized processes managed by the host operating system's kernel , but with a constrained and heavily manipulated view of the system's processes, resources, and environment.
On the host system , a piece of software called a hypervisor is responsible for starting, stopping, and managing the virtual machines.	Containers are unaware that they exist on a shared system and operate as if they were in full control of the computer .
Because VMs are operated as completely distinct computers that, under normal operating conditions, cannot affect the host system or other VMs, virtual machines offer great isolation and security.	it is more common to manage containers more similarly to applications.
In general, virtual machines let you subdivide a machine's resources into smaller, individual computers, but the result doesn't differ significantly from managing a fleet of physical computers.	containers occupy a space that sits somewhere in between the strong isolation of virtual machines and the native management of conventional processes.

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

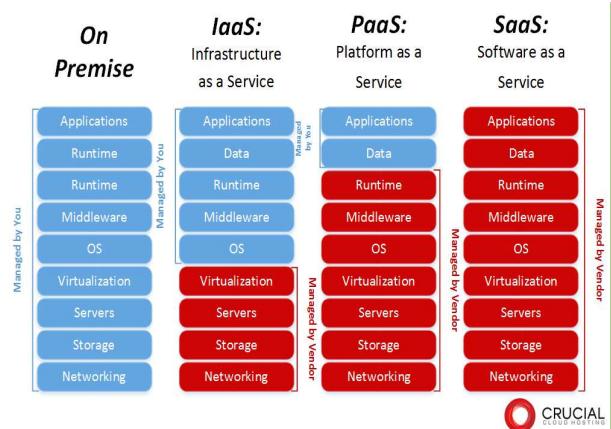


Platform as a Service PaaS

Introducing Platform as a Service

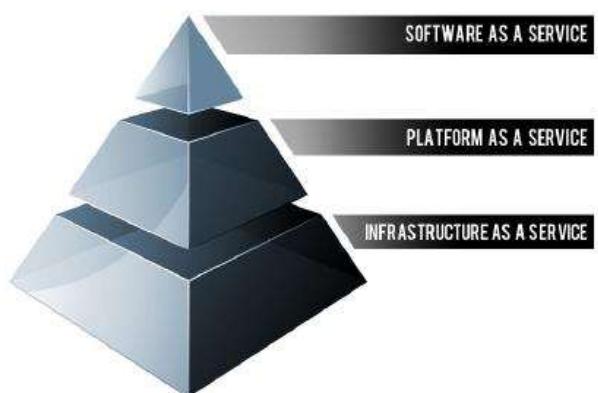
- The capability provided to the consumer is to **deploy** onto the **cloud infrastructure**, **consumer-created** or acquired applications created using programming languages and **tools supported by the provider**.
- The consumer **does not manage** or control the underlying **cloud infrastructure** including network, servers, operating systems, or storage, but has control over the **deployed applications** and possibly application **hosting** environment **configurations**.
- A **PaaS** platform offers an environment on which developers **create** and deploy **applications** and do not necessarily need to know how many **processors** or how much **memory** that **applications** will be using.
- In addition, multiple programming models and **specialized services** (e.g., data access, authentication, and payments) are offered as building blocks to new applications.

Google AppEngine, Azure, Force.com are examples of Platform as a Service



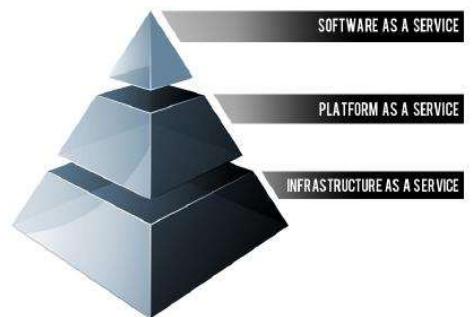
Building Blocks - PaaS

- 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- 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

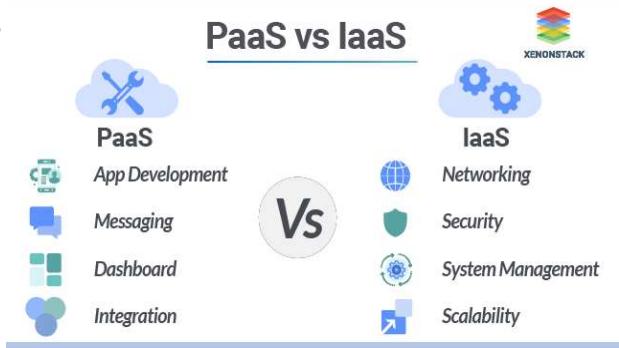


BITS Pilani

PaaS vs IaaS

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;

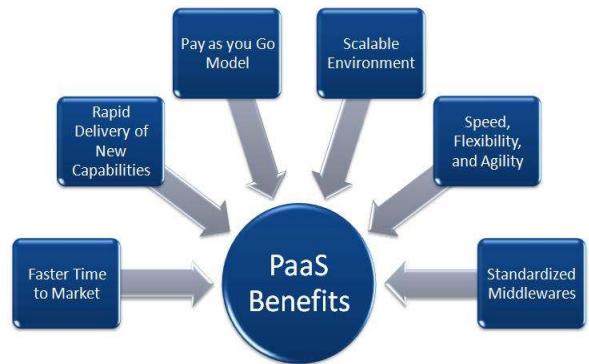
1. A collaborative platform for software development, focused on workflow management regardless of the data source being used for the application. An example of this approach would be Heroku, a PaaS that utilizes the Ruby on Rails development language.
2. A platform that allows for the creation of software 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



PaaS Advantages

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.



PaaS Disadvantages

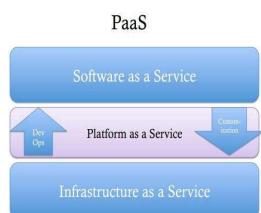
- Since users rely on a provider's infrastructure and software, vendor lock-in can be an issue in PaaS environments.
- Other risks associated with PaaS are provider downtime or a provider changing its development roadmap.
- If a provider stops supporting a certain programming language, users may be forced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



Who Can Use PaaS



- My code – running
 - Not a "VM" but a Virtual App Server
- Not just code
 - I like Queues and Topics, ESB flows, Workflows, Databases, Logs, Portals, etc.
- Not just Runtime
 - I like SVN, Git, build, continuous integration, code coverage, automated test



- Moreover, if you are a manager of a group of developers, you probably like governance.

- Software developers, web developers and businesses can benefit from PaaS.
- **For example,** web developers can use individual PaaS environments at every stage of the process to develop, test and ultimately host their websites. However, businesses that are developing their own internal software can also utilise Platform as a Service, particularly to create distinct ring-fenced development and testing environments.

BITS Pilani

PaaS – Best Practices

Managing PaaS

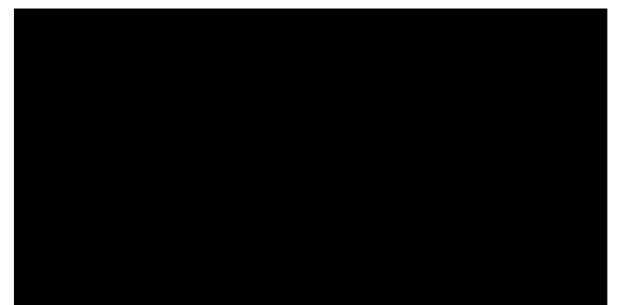
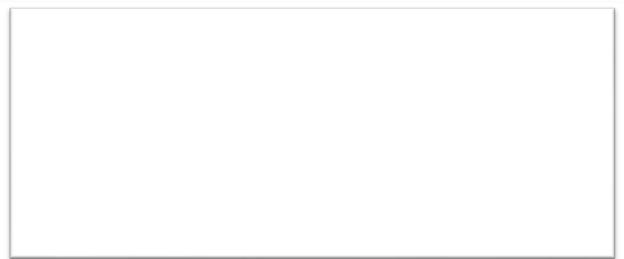
- If you choose to maintain the software yourself, you must set up, configure, maintain, and administer the PaaS yourself (either on a public or private cloud).
- Alternatively, you can have the vendor to provide these services. The result is reduced friction between the development and deployment teams. There will, of course, be situations in which it's critical for the internal team to control and manage a complex software environment.

Best Practices

- Start with the data, and work up to the services and UI. No matter what the PaaS provider suggests.
- Define a staging and testing strategy before you begin development.
- Consider SOA approaches in the design and deployment of the PaaS- bases application.
- Make sure to do load testing along with functional testing.
- Make sure to model performance.
- Don't fall in love with a PaaS player, you may need to use several.

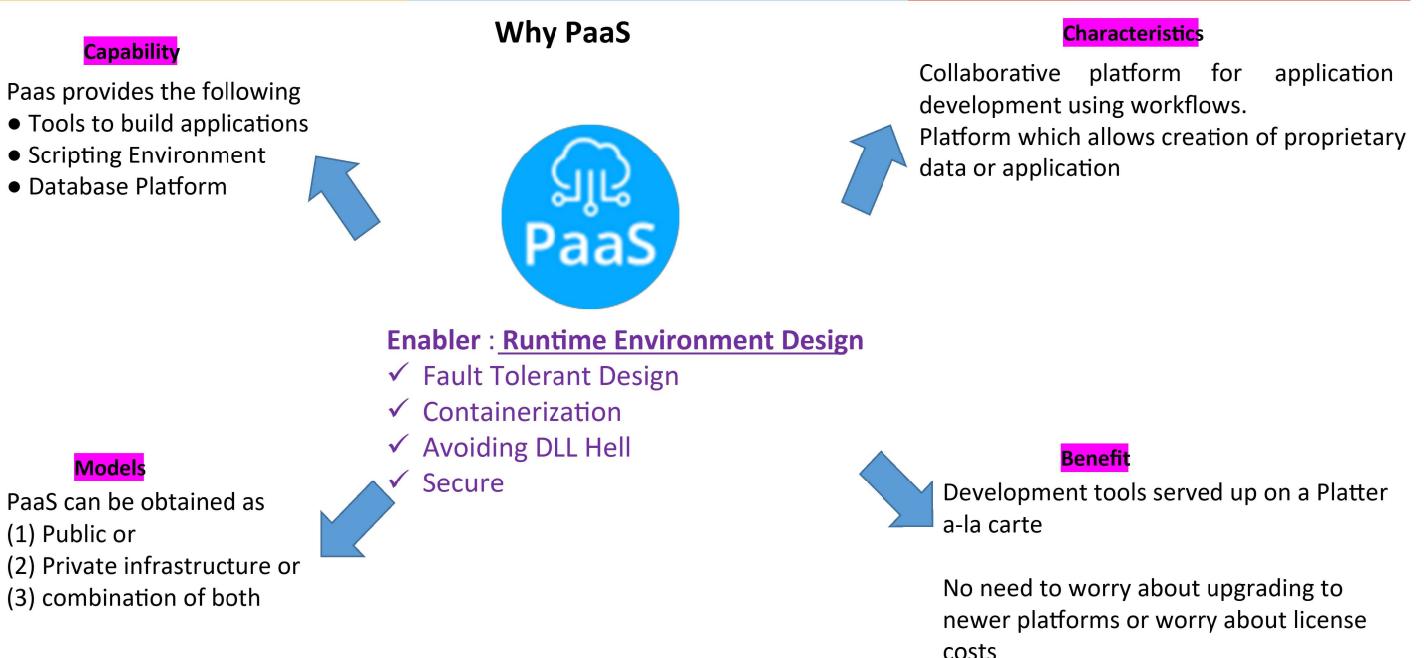
PaaS – AWS Examples

- AWS Lambda
 - Serverless code
 - On demand / respond to events
- AWS Elastic Beanstalk
 - Focus on developing features of your web application
 - Amazon takes care of provisioning resources, scaling, patching etc.



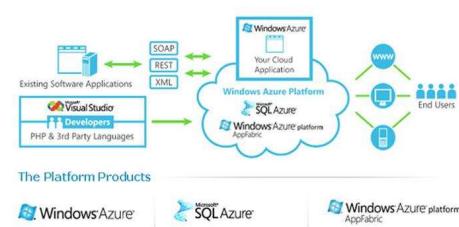
BITS Pilani

Platform as a Service - Summary



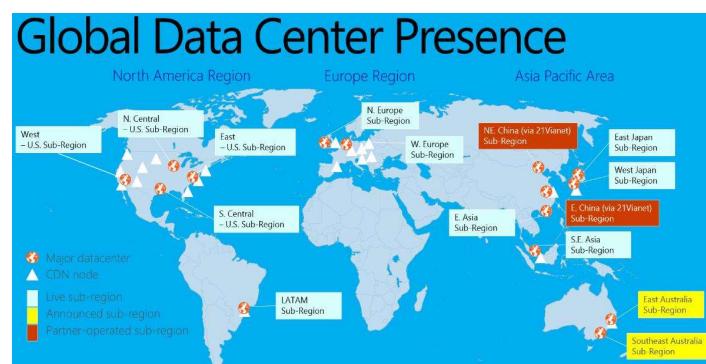


Windows Azure



What is Microsoft Azure?

- **Windows Azure**, which was later renamed as **Microsoft Azure** in 2014, is a **cloud computing platform**, designed by Microsoft to successfully build, deploy, and manage applications and services through a global network of data centers.
- Azure Platform is divided into three major component platforms. **Compute, Storage & Fabric Controller**.
- Azure can be described as the managed data centers that are used to build, deploy, manage the applications and provide services through a global network.
- The services provided by Microsoft Azure are **PaaS** and **IaaS**.
- Many programming languages and frameworks are supported by it. This platform is built over Microsoft data centers.
- Windows Azure can be used to create, distribute and upgrade Web applications without the need to maintain expensive, often underutilized resources onsite



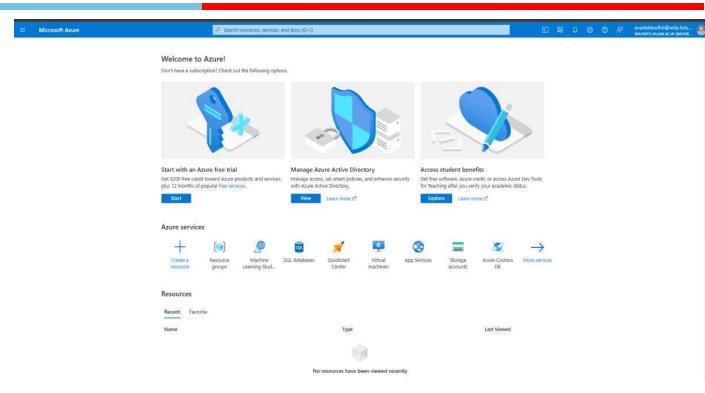
Windows Azure offers a cloud platform built on Windows OS and based on Microsoft virtualization technology.

Azure Components?

Azure provides more than **200 services**, are divided into **18 categories**.

These categories include **computing, networking, storage, IoT, migration, mobile, analytics, containers, artificial intelligence**, and other machine learning, integration, management tools, developer tools, security, databases, DevOps, media identity, and web services.

There are 42 Azure data centers spread around the globe, which is the highest number of data centers for any cloud platform. Also, Azure is planning to get 12 more data centers, which will increase the number of data centers to 54, shortly.



Azure Services?

Compute Services

- **Virtual Machine**

This service enables you to create a virtual machine in Windows, Linux or any other configuration in seconds.

- **Cloud Service**

This service lets you create scalable applications within the cloud. Once the application is deployed, everything, including provisioning, load balancing, and health monitoring, is taken care of by Azure.

- **Service Fabric**

With service fabric, the process of developing a microservice is immensely simplified. Microservice is an application that contains other bundled smaller applications.

- **Functions**

With functions, you can create applications in any programming language. The best part about this service is that you need not worry about hardware requirements while developing applications because Azure takes care of that. All you need to do is provide the code.

Networking

- **Azure CDN**

Azure CDN (Content Delivery Network) is for delivering content to users. It uses a high bandwidth, and content can be transferred to any person around the globe. The **CDN service** uses a network of servers placed strategically around the globe so that the users can access the data as soon as possible.

- **Express Route**

This service lets you connect your on-premise network to the Microsoft cloud or any other services that you want, through a private connection. So, the only communications that will happen here will be between the enterprise network and the service that you want.

- **Virtual network**

The **virtual network** allows you to have any of the Azure services communicate with one another privately and securely.

- **Azure DNS**

This service allows you to host your DNS domains or system domains on Azure.

Azure Services?

Storage

- Disk Storage

This service allows you to choose from either HDD (Hard Disk Drive) or SSD (Solid State Drive) as your storage option along with your virtual machine.

- Blob Storage

This service is optimized to store a massive amount of unstructured data, including text and even binary data.

- File Storage

This is a managed file storage service that can be accessed via industry SMB (server message block) protocol.

- Queue Storage

With queue storage, you can provide stable message queuing for a large workload. This service can be accessed from anywhere in this world.

Azure key Terms?

Concept Name	Description
Regions	Azure is a global cloud platform which is available across various regions around the world. When you request a service, application, or VM in Azure, you are first asked to specify a region. The selected region represents datacenter where your application runs.
Datacenter	In Azure, you can deploy your applications into a variety of data centers around the globe. So, it is advisable to select a region which is closer to most of your customers. It helps you to reduce latency in network requests.
Azure portal	The Azure portal is a web-based application which can be used to create, manage and remove Azure resource and services. It is located at https://portal.azure.com .
Resources	Azure resource is an individual computer, networking data or app hosting services which charged individually. Some common resources are virtual machines(VM), storage account, or SQL databases.
Resource groups	An Azure resource group is a container which holds related resource for an Azure solution. It may include every resource or just resource which you wants to manage.
Resource Manager templates	It is a JSON which defines one or more resource to deploy to a resource group. It also establishes dependencies between deployed resources.
Automation:	Azure allows you to automate the process of creating, managing and deleting resource by using PowerShell or the Azure command-line Interface(CLI).
Azure PowerShell	PowerShell is a set of modules that offer cmdlets to manage Azure. In most cases, you are allowed to use, the cmdlets command for the same tasks which you are performing in the Azure portal.
Azure command-line interface(CLI)	The Azure CLI is a tool that you can use to create, manage, and remove Azure resources from the command line.

Azure PaaS Design Principles

Ten design principles for Azure applications

Article • 07/19/2022 • 2 minutes to read • 11 contributors



Follow these design principles to make your application more scalable, resilient, and manageable.

- **Design for self healing.** In a distributed system, failures happen. Design your application to be self healing when failures occur.
- **Make all things redundant.** Build redundancy into your application, to avoid having single points of failure.
- **Minimize coordination.** Minimize coordination between application services to achieve scalability.
- **Design to scale out.** Design your application so that it can scale horizontally, adding or removing new instances as demand requires.
- **Partition around limits.** Use partitioning to work around database, network, and compute limits.
- **Design for operations.** Design your application so that the operations team has the tools they need.
- **Use managed services.** When possible, use platform as a service (PaaS) rather than infrastructure as a service (IaaS).
 - **Use an identity service.** Use an identity as a service (IDaaS) platform instead of building or operating your own.
- **Use the best data store for the job.** Pick the storage technology that is the best fit for your data and how it will be used.
- **Design for evolution.** All successful applications change over time. An evolutionary design is key for continuous innovation.
- **Build for the needs of business.** Every design decision must be justified by a business requirement.



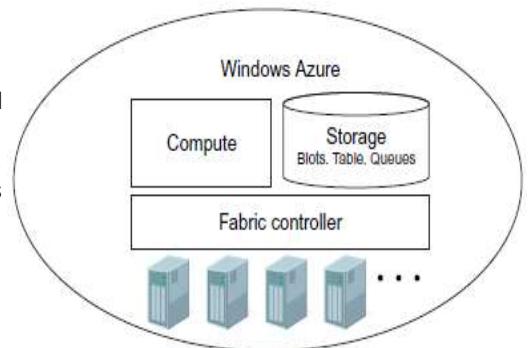
BITS Pilani

Azure Introduction

- Windows Azure provides a **platform to develop applications** using a range of available technologies and programming languages.
- It offers to create and deploy applications using .net platform, which is Microsoft's own application development technology. In addition to .net, there are many more technologies and languages supported. For example, Java, PHP, Ruby, Oracle, Linux, MySQL, Python.
- Windows Azure applications are scaled by creating **multiple instances** of the application.
- The **number of instances** needed by the **application is specified by the developer** while hosting the applications.
- If traffic is increased or decreased on the website or web application, it can be managed easily by logging in to Windows Azure management portal and specifying the instances. Load balancing can also be automated which would allow Azure to make the decision itself as when to assign more resources to application.
- Web applications support .net, java, python, php and node.js. Tasks such as scaling, and backups can be easily automated.
- A new feature called 'webjobs' is available, which is a kind of batch processing service. Webjobs can also be scaled and scheduled.
- The mobile application platforms supported are Xamarin iOS, Xamarin Android and IOS.
- Azure platform is developed in such a way that developers need to **concentrate on only the development part** and need not **worry about other technical stuff outside their domain**. Thus most of the administrative work is done by Azure itself.

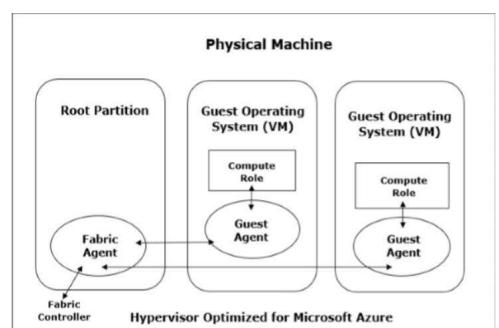
Azure Runtime Environment

- The Windows Azure runtime environment provides a scalable compute and storage hosting environment along with management capabilities. It has three major components: Compute, Storage and the Fabric Controller
- The hosting environment of Azure is called the **Fabric Controller**. It has a pool of individual systems connected on a network and automatically manages resources by load balancing and geo-replication. It manages the application lifecycle without requiring the hosted apps to explicitly deal with the scalability and availability requirements. Each physical machine hosts an Azure agent that manages the machine.
- The **Azure Compute Service** provides a Windows-based environment to run applications written in the various languages and technologies supported on the Windows platform.
- The Windows **Azure storage service** provides scalable storage for applications running on the Windows Azure in multiple forms. It enables storage for binary and text data, messages and structured data through support for features called Blobs, Tables, Queues and Drives.



Azure fabric Controller

- Fabric Controller is a significant part of Windows Azure architecture.
- Inside the data centre, there are many machines or servers aggregated by a switch. We can say that fabric controller is a brain of the azure service that analyses the processes and makes decisions.
- Fabrics** are group of machines in Microsoft's data centre which are aggregated by a switch. The group of these machines is called **cluster**.
- Each cluster is managed and owned by a fabric controller. They are replicated along with these machines. It manages everything inside those machines, for e.g., load balancers, switches, etc. Each machine has a **fabric agent** running inside it and fabric controller can communicate with each fabric agent
- When a user chooses one of the virtual machine, the operating system, patch updates and software updates are performed by fabric controller. It decides where the new application should run which is one of the most important functions of Fabric Controller. It also selects the physical server to optimize hardware utilization
- When a new application is published in Azure, an application configuration file written in XML is also attached. The fabric controller reads those files in Microsoft datacenter and makes the setting accordingly



Imagine a situation where **four instances of web role** are running, and one of them dies.

The **fabric controller** will initiate a **new instance** to replace the dead one immediately.

Similarly, in case **any virtual machine fails**, a **new one is assigned by the fabric controller**. It also **resets the load balancers** after **assigning the new machine**, so that it points to the new machine instantaneously.

Thus, all the intelligent tasks are performed by the Fabric Controller in Windows Azure architecture.

Azure Components

- When the **system is running**, services are **monitored and one can access event logs, trace/ debug data, performance counters, IIS web server logs, crash dumps, and other log files.**
- **This information can be saved in Azure storage.** Note that there is **no debugging capability** for running cloud applications, but **debugging is done from a trace.**
- Like most PaaS services, Windows Azure defines a programming **model** specific to the platform, which is called the **Web role- Worker role model**.
- **Cloud Service Role:** In Azure, a Cloud Service Role is a collection of managed, load-balanced, Platform-as-a-Service virtual machines that work together to perform common tasks. Cloud Service Roles are managed by **Azure fabric controller** and provide the **ultimate combination of scalability, control, and customization**
- **Web Role** is a Cloud Service role in Azure that is configured and customized to **run web applications developed on programming languages / technologies that are supported by Internet Information Services (IIS)**, such as ASP.NET, PHP, Windows Communication Foundation and Fast CGI
- **Worker Role** is any role in Azure that **runs applications and services level tasks**, which generally do not **require IIS**. In Worker Roles, IIS is not installed by default. They are mainly used to **perform supporting background processes** along with Web Roles and do tasks such as automatically compressing uploaded images, run scripts when something changes in database, get new messages from queue and process and more.

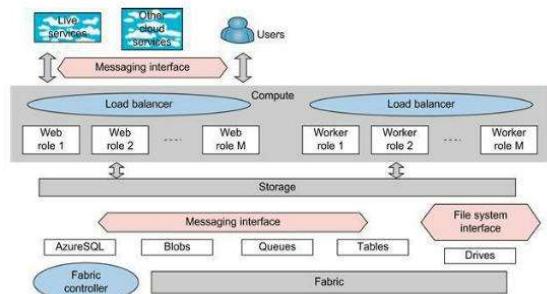
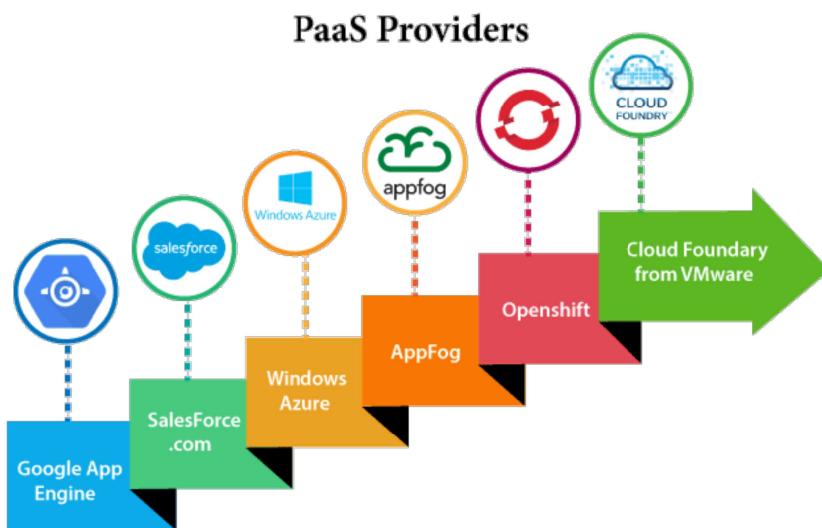


FIGURE 6.25 Features of the Azure cloud platform.

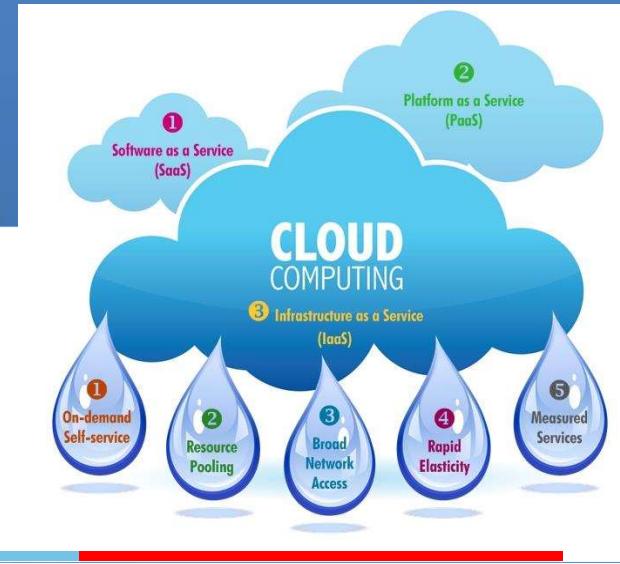
BITS Pilani

PaaS – Vendors (Popular)



Common PaaS vendors include Salesforce.com's Force.com, which provides an enterprise customer relationship management (CRM) platform. PaaS platforms for software development and management include Appear IQ, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine and Heroku.

Common PaaS vendors include Salesforce.com's Force.com, which provides an enterprise customer relationship management (CRM) platform. PaaS platforms for software development and management include Appear IQ, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine and Heroku.



Software as a Service

Agenda



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

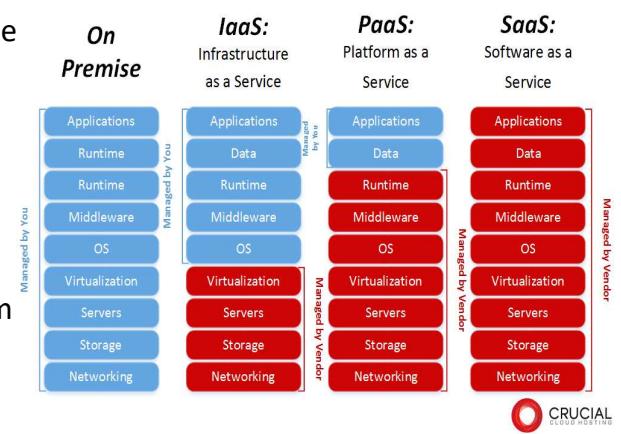
Introducing Software as a Service

• Software as a service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet.

• Shortly, in the SaaS model software is deployed as a hosted service and accessed over the Internet, as opposed to “On Premise.”

• Software delivered to home consumers, small business, medium and large business

The traditional model of software distribution, in which software is purchased for and installed on personal computers, is sometimes referred to as software as a product.



BITS Pilani

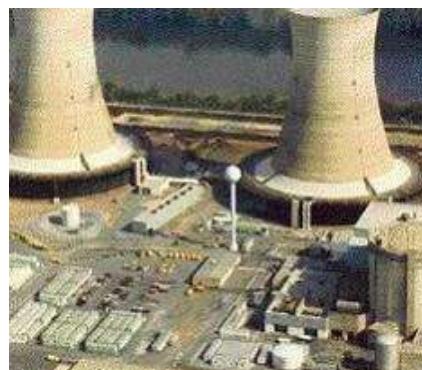
SaaS Motivations

• In the traditional model of software delivery, the customer acquires a perpetual license and assumes responsibility for managing the software.

• There is a high upfront cost associated with the purchase of the license, as well as the burden of implementation and ongoing maintenance.

• ROI is often delayed considerably, and, due to the rapid pace of technological change, expensive software solutions can quickly become obsolete.

Traditional Software



On-Demand Utility



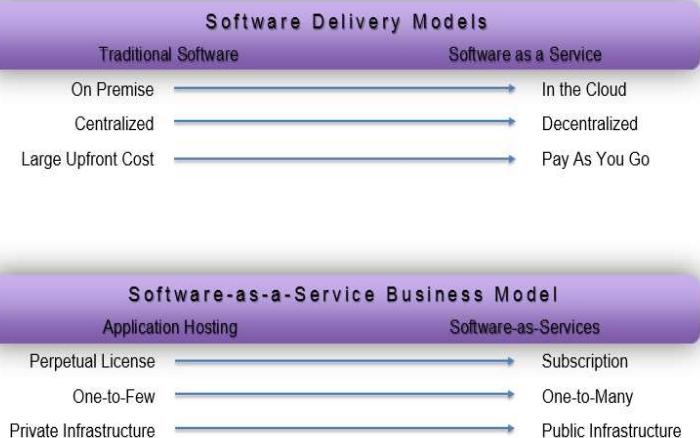
Build Your Own

Plug In, Subscribe
Pay-per-Use

SaaS Delivery Model

- The web as a platform is the centre point. The web as a platform is the centre point
- Network-based access to, and management of, commercially available (i.e., not custom) software application delivery that typically is closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics
- Software delivered to home consumers, small business, medium and large business

§ The traditional model of software distribution, in which software is purchased for and installed on personal computers, is sometimes referred to as software as a product.



BITS Pilani

SaaS Architecture

Run by

- Bandwidth technologies
- The cost of a PC has been reduced significantly with more powerful computing but the cost of application software has not followed
- Timely and expensive setup and maintenance costs
- Licensing issues for business are contributing significantly to the use of illegal software and piracy.

Scalable

- Multitenant efficient
- Configurable

Scaling the application - maximizing concurrency, and using application resources more efficiently

- i.e. optimizing locking duration, statelessness, sharing pooled resources such as threads and network connections, caching reference data, and partitioning large databases.

SaaS Architecture

Multi-tenancy – important architectural shift from designing isolated, single-tenant applications

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

Configurable - a single application instance on a single server has to accommodate users from several different companies at once

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

BITS Pilani

SaaS Model Comparison

Traditional	Software as a Service
Designed for customers to install, manage and maintain.	Designed from the outset up for delivery as Internet-based services.
Architect solutions to be run by an individual company in a dedicated instantiation of the software.	Designed to run thousands of different customers on a single code.
Infrequent, major upgrades every 18-24 months, sold individually to each installed base customer.	Frequent, "digestible" upgrades every 3-6 months to minimize customer disruption and enhance satisfaction.
Version control Upgrade fee	Fixing a problem for one customer fixes it for everyone
Streamlined, repeatable functionality via Web services, open APIs and standard connectors	May use open APIs and Web services to facilitate integration, but each customer must typically pay for one-off integration work.

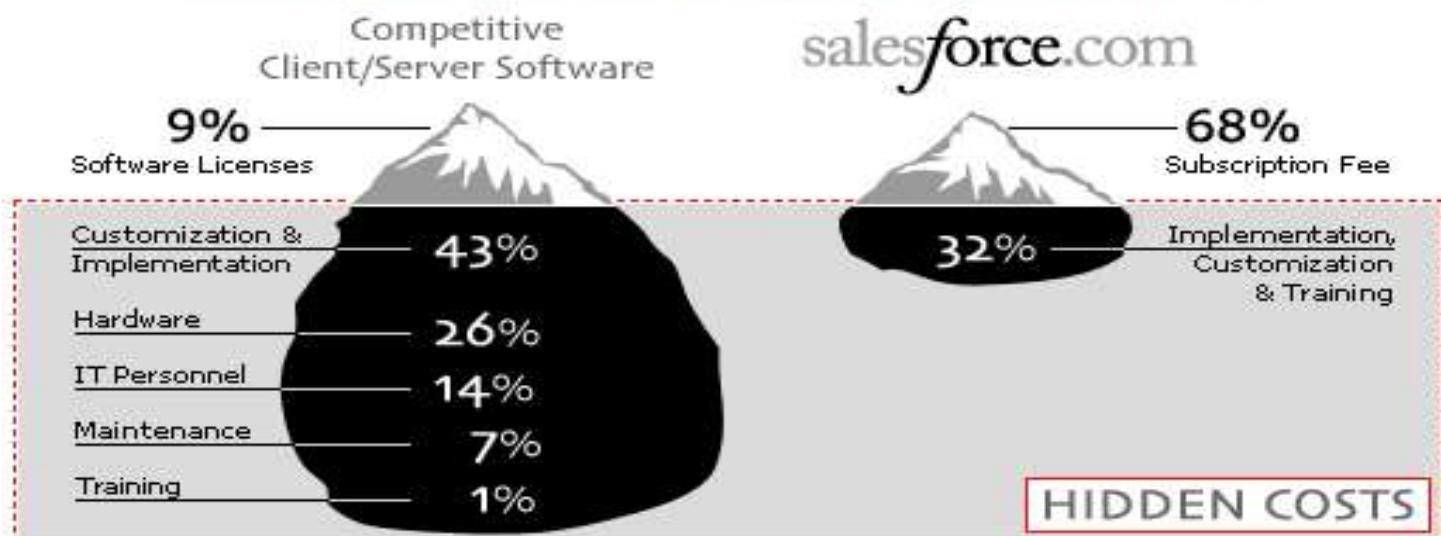
SaaS Model Comparison

Attributes	Software-as-a-Service (SaaS)	On-Premise
Alternate labels	On-Demand, Subscription Based, Hosted, Application Service Provider (ASP)	Installed, Hosted On-Premise
Purchase model	Lease, rent, subscription	Most commonly purchasing of licenses (ownership) with some lease and rent options
Maintenance and support	Typically included	For purchase option it would be based on the percentage of license fees
Security	Typically in a 3rd party highly secure data center	Responsibility of the customer
Upgrades	Typically included	Included with maintenance
Data model	Shared (multi-tenant) or dedicated (single) instance depending on the vendor	Dedicated (single) instance on the customer's servers
Access	Typically all major web browsers	Typically all major web browsers
Initial investment	Low upfront cost since no hardware or infrastructure investment	Higher cost since typically purchasing licenses and hardware
3-5 year investment	Reoccurring fee	One-time fee
Legal implications	Hosted by 3rd party and lack of ownership. This needs to be clearly defined in a SLA	Less of an issue with the purchase option where customer claims ownership
Implementation	Typically shorter since all is hosted by the vendor	Can be lengthier since it is installed and integrated with the existing infrastructure
Integration	Can be limited due to Web standards and protocols	Tends to be more flexible since it resides behind your firewall on your servers

BITS Pilani

SaaS Model Comparison

Avoid the hidden costs of traditional CRM software



SaaS Adantages

Characteristics	Benefits
Network delivered access to commercially available software	No local infrastructure or software to purchase or maintain Applications & data are available anywhere with network connectivity
Application delivery is one-to-many model	Operating costs are reduced by managing infrastructure in central locations rather than at each customer's site
Built on optimized & robust platform	Improved availability and reliability
Customer pays for as much as they need when they need it	Lower TCO

VIRTUAL OFFICE



E-MAIL COMMUNICATION



STORAGE



BITS Pilani

SaaS Providers

1. Lumen5	2. FutureFuel	3. Squibler	8. Cloud-Based Microsoft Office 365	9. Salesforce	10. G Suite
4. Buffer	5. Dropbox	6. AWS	11. ZenDesk	12. Visme	13. Slack
	7. HubSpot		14. Canva	15. Box	

SaaS Providers at a Glance

SaaS Providers

<p>Lumen5 is a leading video creator SaaS app that lets businesses create amazing videos with its drag-and-drop interface. It creates video automatically from text or any URL. The text is converted into a video that can be personalized by positioning text, adding images from the library, highlighting key words with brand colors, tweaking font style, and changing video resolution. Lumen5 receives 220K+ monthly visitors which shows how popular it is.</p> <p>FutureFuel is a perfect SaaS example that is sure to inspire you. It is a student debt management app for businesses and HR professionals. Companies attract and retain top talent by addressing repaying the loans of students. The students, in return, work for their companies and help employees. FutureFuel provides businesses with access to the future workforce that will stick with them for years to come.</p> <p>Squibler is a SaaS creativity app that helps writers tell stories. It has a great interface that makes writing hassle-free. It has many prompt writing tools that serve all types of writing such as books, novels, journals, screenwriting, or others.</p> <p>Buffer is a perfect SaaS example for pretty much any SaaS business out there. It is a simple yet effective tool that helps social media management scheduling easier for businesses of all sizes. But what's more important is that how Leo Widrich guest blogged for 10 months which helped Buffer reach 100K users in less than a year.</p>
--

BITS Pilani

SaaS Providers at a Glance

SaaS Providers

<p>Dropbox is a leading cloud storage SaaS company that makes it easier for businesses to store, share, and collaborate on files and data on the go. It is a smart workplace that lets your workforce work from anywhere.</p> <p>Amazon Web Services (AWS) is a SaaS example. With more than 150 services on offer, AWS provides businesses and individuals with all the tools they need such as database, IoT, business applications, machine learning, storage, robotics, security, customer engagement, blockchain, and more.</p> <p>HubSpot is a leading SaaS tool for businesses and even enterprise software companies. It offers sales, marketing, CRM, CMS, and services software for businesses. The marketing software helps businesses attract visitors, convert them, close deals, and retain them.</p> <p>Cloud-based Office 365 is indeed something worth talking about. You can now create, edit, share, manage, and access your office files from anywhere. Office 365 cloud is a pure SaaS example that shows how Microsoft has fulfilled the needs of its users by offering them SaaS products. You can no longer use Microsoft office 365 on your personal computer but with the cloud platform, you get access to it on the go as you can access it from any Microsoft data center.</p>

BITS Pilani

SaaS Providers at a Glance

SaaS Providers

Salesforce It is one of the most popular software as a service provider that excels in cloud computing. It is a complete customer relationship management suite for businesses. Salesforce lets businesses collect, store, access, monitor, and analyze customer data from a single dashboard.

The **G Suite** is the Google Cloud SaaS applications that include several cloud-based apps. G Suite includes Gmail, Calendar, Hangouts, Google Sheets, Docs, Forms, Slides, Sites, Vault, and several other apps.

ZenDesk is another example of a SaaS company that is indeed inspiring. It is a SaaS company with annual revenue of more than \$5 million. It is a customer support and ticketing software that supports several support channels including phone, email, like chat, social media, online tickets, and more.

Box is a cloud content management and file-sharing that is publicly traded SaaS company. It is a multi-purpose software for businesses that let users collaborate, automate, share, and manage content and files over the cloud. It supports secure file sharing and files can be accessed from desktop and web. Team members can collaborate and discuss everything in real-time on the document they are working on.

BITS Pilani



SaaS User Benefits

- **Lower Cost of Ownership**

- The software is paid when it is consumed, no large upfront cost for a software license. [Salesforce.com](#) has a best-of-breed CRM system for \$59.00 per user per month, with no upfront.
- Since no hardware infrastructure, installation, maintenance, and administration, budgeting is easy
- The software is available immediately upon purchasing

- **Focus on Core Competency**

- The IT saving on capital and effort allows the customer to remain focused on their core competency and utilize resources in more strategic areas.

- **Access Anywhere**

- Users can use their applications and access their data anywhere they have an Internet connection and a computing device
- This enhances the customer experience of the software and makes it easier for users to get work done fast

- **Freedom to Choose (or Better Software)**

- The pay-as-you-go (PAYG) nature of SaaS enables users to select applications they wish to use and to stop using those that no longer meet their needs. Ultimately, this freedom leads to better software applications because vendors must be receptive to customer needs and wants.



SaaS User Benefits

- **New Application Types**

- Since the barrier to use the software for the first time is low, it is now feasible to develop applications that may have an occasional use model. This would be impossible in the perpetual license model. If a high upfront cost were required the number of participants would be much smaller.

- **Faster Product Cycles**

- Product releases are much more frequent, but contain fewer new features than the typical releases in the perpetual license model because the developer know the environment the software needs to run
- This new process gets bug fixes out faster and allows users to digest new features in smaller bites, which ultimately makes the users more productive than they were under the previous model.

BITS Pilani

SaaS Vendor Benefits

- **Increased Total Available Market**

- Lower upfront costs and reduced infrastructure capital translate into a much larger available market for the software vendor, because users that previously could not afford the software license or lacked the skill to support the necessary infrastructure are potential customers.
- A related benefit is that the decision maker for the purchase of a SaaS application will be at a department level rather than the enterprise level that is typical for the perpetual license model. This results in shorter sales cycles.

- **Enhanced Competitive Differentiation**

- The ability to deliver applications via the SaaS model enhances a software company's competitive differentiation. It also creates opportunities for new companies to compete effectively with larger vendors.
- On the other hand, software companies will face ever-increasing pressure from their competitors to move to the SaaS model.
- Those who lag behind will find it difficult to catch up as the software industry continues to rapidly evolve.

- **Lower Development Costs & Quicker Time-to-Market**

- The main saving is at testing (35%). Small and frequent releases – less to test
 - Application is developed to be deployed on a specific hardware infrastructure, far less number of possible environments – less to test
 - This, in turn, provides the software developer with overall lower development costs and quicker time-to-market.



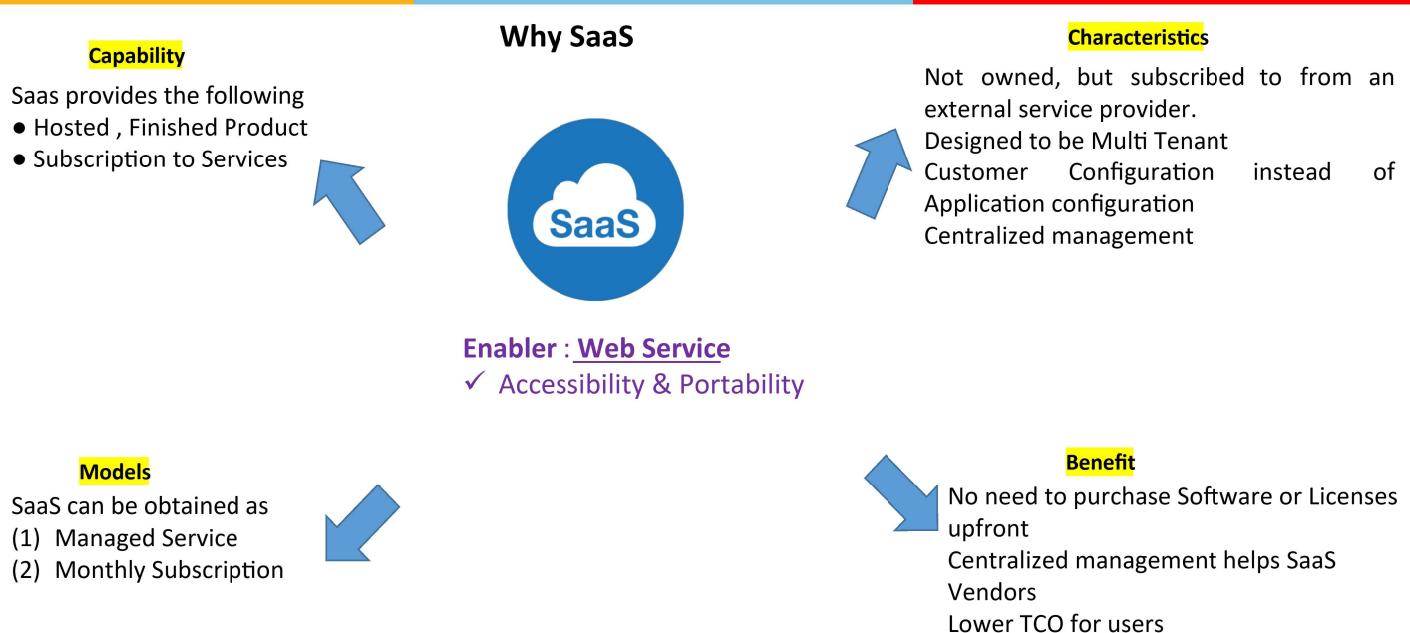
SaaS Vendor Benefits

- **Effective Low Cost Marketing**
 - Between 1995 and today, buyers' habits shifted from an outbound world driven by field sales and print advertising to an inbound world driven by Internet search.
- **Predictable MRR Revenue**
 - Traditionally, software companies rely on one major release every 12-18 months to fuel a revenue stream from the sale of upgrades (long tail theory).
 - In the SaaS model the revenue is typically in the form of Monthly Recurring Revenue (MRR)
- **Improved Customer Relationships**
 - SaaS contributes to improved relationships between vendors and customers.
- **Protecting of IP**
 - Difficult to obtain illegal copies
 - Price is low, making getting an illegal copies totally unnecessary



BITS Pilani

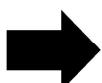
Software as a Service



SaaS Applicability Scenarios

1 Single-User software application

- Organize personal information
- Run on users' own local computer
- Serve only one user at a time
- **Inapplicable to SaaS model**
 - Data security issue
 - Network performance issue
- Example: Microsoft office suite (Prior to 0365)



2 Infrastructure software

- Serve as the foundation for most other enterprise software application
- **Inapplicable to SaaS model**
- Installation locally is required
- Form the basis to run other application
- Example: Window XP, Oracle database

3 Embedded Software

Software component for embedded system
Support the functionality of the hardware device
Inapplicable to SaaS model
Embedded means software and hardware is combined together and is inseparable like a bios / firmware
Example: software embedded in ATM machines, cell phones, routers, medical equipment, etc



BITS Pilani

The Right SaaS Model?

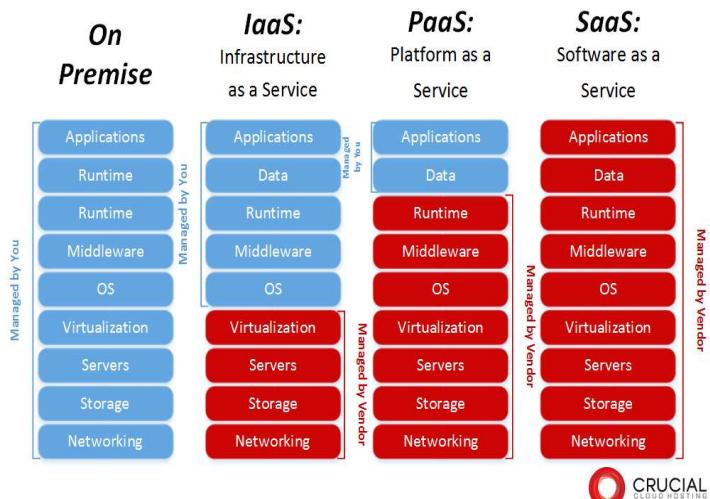


• Enterprise Software Application

- Perform business functions
- Organize internal and external information
- Share data among internal and external users
- The most standard type of software applicable to SaaS model
- Example: Salesforce.com CRM application, Siebel On-demand application

PARAMETER	SAAS	PAAS	IAAS
Full Form	Software As a Service	Platform as a Service	Infrastructure as a Service
General Users	Business Users	Developers and Deployers	System managers
Services Available	Email , Office automation , CRM , website testing , Virtual desktop	Service and application test , development , integration and deployment	Virtual machines, operating systems, network, storage, backup services.
Business Justification	To complete business tasks	Create and deploy service and applications for users	Create platform for service and application test, development.
Examples	Paypal , Salesforce.com	Azure Service platform, Force.com	Amazon EC2 , GoGrid
Control	Highest degree of control and flexibility	Good degree of control and flexibility	Minimal degree of control and flexibility
Operational Cost	Minimal	Lower	Highest
Portability	No portability	Lower	Best
Risk Of Vendor Interlock	Highest	Medium	Lowest
Security	Requires transparency in service provider's security policies to be able to determine the degree of sensitive corporate data.	Additional security is required to make sure rogue applications don't exploit vulnerabilities in software platform.	Should consider Virtual and physical servers security policy conformity.

SaaS vs PaaS vs IaaS



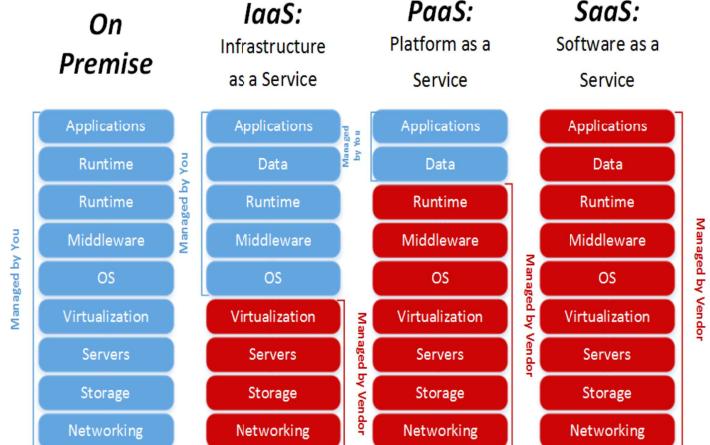
CRUCIAL
CLOUD HOSTING

COMPARISON

BITS Pilani

Course area	IaaS	PaaS	SaaS
Cloud computing	<ul style="list-style-type: none"> - Virtualization - Software defined networks - Software defined data centres - Cloud storage 	<ul style="list-style-type: none"> - Cloud based simulation - Development of cloud software - Windows Azure, Amazon) 	<ul style="list-style-type: none"> - Communication apps - Cloud storage apps - Social computing apps - Apps management - Web hosting service
Mobile technologies	<ul style="list-style-type: none"> - Software defined radio networks 	<ul style="list-style-type: none"> - SMS API - Mobile application development - Mobile agents - Mobile cloud applications 	<ul style="list-style-type: none"> - Mobile commerce apps - M-payment apps - Mobile learning apps - Mobile social apps
Internet of Things	<ul style="list-style-type: none"> - Software defined wireless sensor networks - Smart environments 	<ul style="list-style-type: none"> - API for accessing the sensor data - API for context-aware applications - API for wearable computing applications 	<ul style="list-style-type: none"> - Software for smart device management
Big data	<ul style="list-style-type: none"> - Environment for Hadoop projects - Environment for MongoDB projects 	<ul style="list-style-type: none"> - Map-reduce API 	<ul style="list-style-type: none"> - Data analysis - Visualization
IT management	<ul style="list-style-type: none"> - Cloud management 	<ul style="list-style-type: none"> - Salesforce PaaS - Heroku PaaS 	<ul style="list-style-type: none"> - Project management software - CRM software
Computer simulation and virtual reality	<ul style="list-style-type: none"> - Resources for simulation execution - Environment for rendering 	<ul style="list-style-type: none"> - API for developing 3D models 	<ul style="list-style-type: none"> - Web simulation software - 3D modelling software tools

SaaS vs PaaS vs IaaS



CRUCIAL
CLOUD HOSTING

APPLICABILITY

Agenda



❖ Cloud Security

2

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



Security in the Cloud



Objective

- Cloud Ecosystem : A massive scale of Virtual Resources
 - Also a massive concentration of risk
 - expected loss from a single breach can be significantly larger
 - concentration of "users" represents a concentration of threats
- "Ultimately, you can outsource responsibility but you can't outsource accountability."



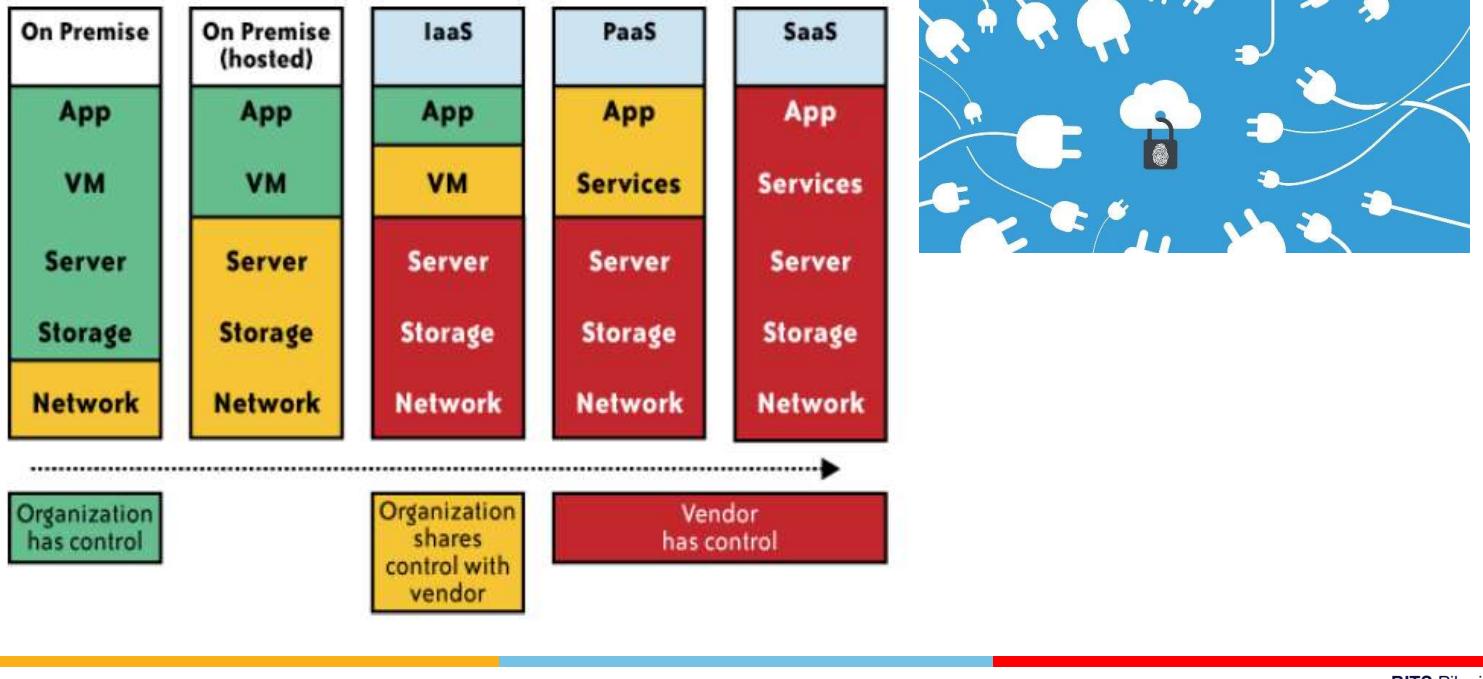
BITS Pilani

Analysis of Security Need

- Cloud computing definitely makes sense if your own security is weak, missing features, or below average.
- Ultimately, if
 - the cloud provider's security people are "better" than yours (and leveraged at least as efficiently),
 - the web-services interfaces don't introduce too many new vulnerabilities, and
 - the cloud provider aims at least as high as you do, at security goals,
 - then cloud computing has better security.



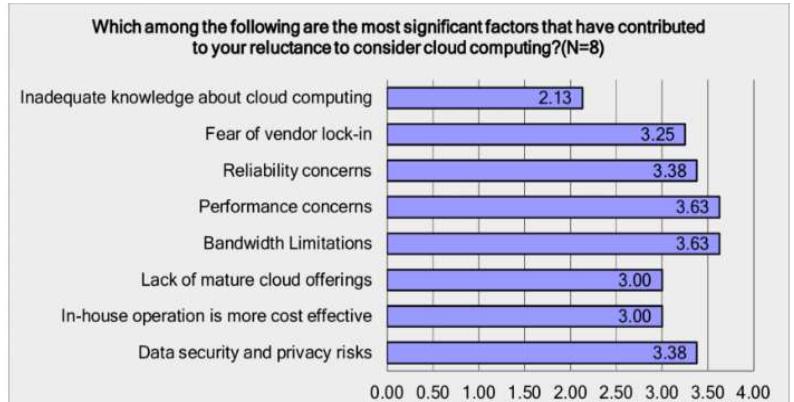
Cloud Governance (L to R)



BITS Pilani

Why Reluctance ?

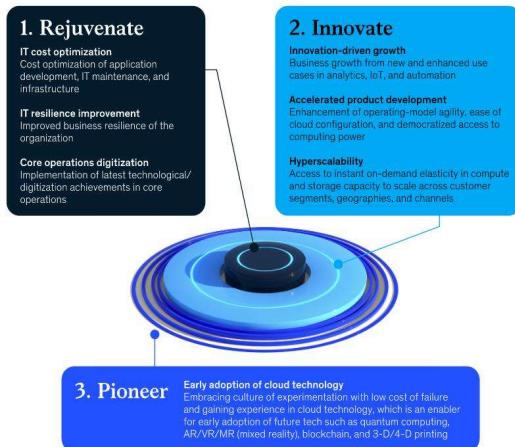
- The cloud acts as a big black box, nothing inside the cloud is visible to the clients
- Clients have no idea or control over what happens inside a cloud
- Even if the cloud provider is honest, it can have malicious system admins who can tamper with the VMs and violate confidentiality and integrity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



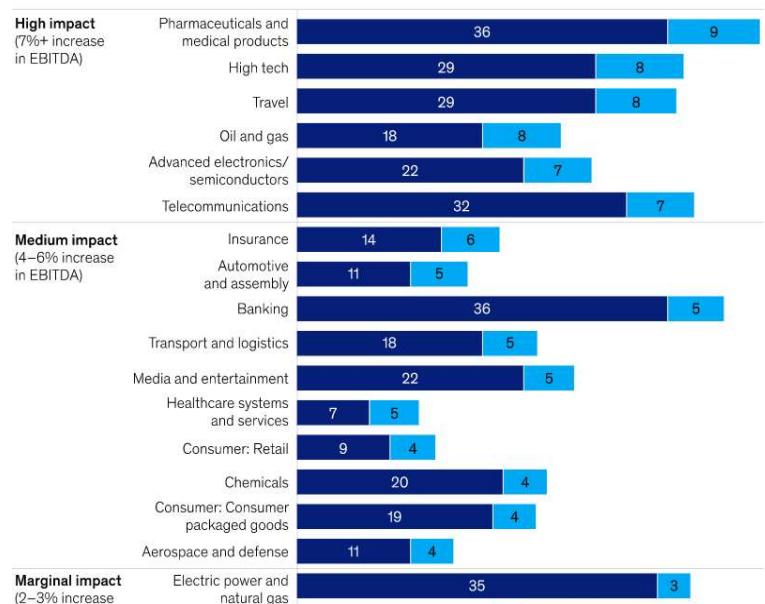
What the Future Holds?

- Large enterprises are getting serious about adopting cloud. They aspire to have roughly 60 percent of their environment in the cloud by 2025.

Cloud value can be captured across three dimensions.



Projected 2030 average EBITDA margin for US industries, %



BITS Pilani

Cloud Security Issues

- Most security problems stem from:
 - Loss of control
 - Lack of trust (mechanisms)
 - Multi-tenancy
- These problems exist mainly in 3rd party management models
- Self-managed clouds still have security issues, but not related to above

Loss of Control

- 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

Trust Issues

- Trusting a third party requires taking risks
 - Defining trust and risk
 - Opposite sides of the same coin (J. Camp)
 - People only trust when it pays (Economist's view)
 - Need for trust arises only in risky situations
-
- Defunct third party management schemes
 - Hard to balance trust and risk
 - e.g. Key Escrow (Clipper chip)
 - Is the cloud headed toward the same path?

BITS Pilani

MT Issues in 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' ?

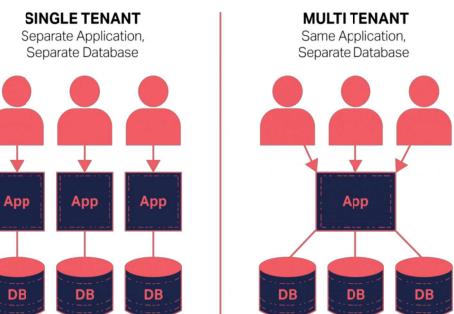
If they can't, can we isolate them?

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?

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



BITS Pilani

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 Attackers can now target the communication link between cloud provider and client

Cloud provider **employees can be phished**

Auditability and **forensics** (out of control of data)

Difficult to audit data held outside organization in a cloud

Forensics also made difficult since now clients don't maintain data locally

Legal dilemma 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 analyzing a security problem, design mitigation strategies, and evaluate solutions

Steps:

- Identify attackers, assets, threats and other components
- Rank the threats
- Choose mitigation strategies
- Build solutions based on the strategies



Basic components

Attacker modeling

Choose what attacker to consider
insider vs. outsider?
single vs. collaborator?

Attacker motivation and capabilities

Attacker goals

Vulnerabilities / threats

Threat Sources

The core issue here is the levels of trust

- Many cloud computing providers trust their customers
- Each customer is physically co-mingling its data with data from anybody else using the cloud while logically and virtually you have your own space
- The way that the cloud provider implements security is typically focused on the fact that those outside of their cloud are evil, and those inside are good.

But what if those inside are also evil?

At client

Learn passwords/authentication information

Gain control of the VMs

At cloud provider

Log client communication

Can read unencrypted data

Can possibly peek into VMs, or make copies of VMs

Can monitor network communication, application patterns

Why?

Gain information about client data

Gain information on client behavior

Sell the information or use itself

Data Security and Storage

Several aspects of data security, including:

Data-in-transit

Confidentiality + integrity using secured protocol

Confidentiality with non-secured protocol and encryption

Data-at-rest

Generally, not encrypted , since data is commingled with other users' data

Encryption if it is not associated with applications?

But how about indexing and searching?

Processing of data, including multitenancy

For any application to process data

BITS Pilani

What is Privacy?

The concept of privacy varies widely among (and sometimes within) countries, cultures, and jurisdictions.

It is shaped by public expectations and legal interpretations;

as such, a concise definition is elusive if not impossible.

Privacy rights or obligations are related to the collection, use, disclosure, storage, and destruction of personal data

At the end of the day, privacy is about the accountability of organizations to data subjects, as well as the transparency to an organization's practice around personal information.



Security in the Cloud – Possible Solutions



Security Issues in the Cloud – The What

In theory, minimizing any of the issues would help third Party Cloud Computing

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)

Multi-tenancy

Private cloud

Takes away the reasons to use a cloud in the first place

VPC: its still not a separate system

Strong separation

Security Issues in the Cloud – My Diagnosis

CLOUD SECURITY ISSUES DETECTION DOES NOT MINIMIZE THE RISK, YOU NEED TO REMEDY.

BITS Pilani

Security Issues in the Cloud – Remedy

Contemporary ecosystem, comprises of several bespoke or COTS solution which can proactively monitor your cloud workloads. They can be grouped into the following

Cloud Workload Protection Platform (CWPP) : These are tools or services which proactively scans workload against known vulnerabilities. Similar to any kind of AV we have on our PC's

Cloud Security Posture Management (CSPM) : These are tools or services focused on finding infrastructure misconfigurations, like a resource with no admission control rule, or policy violation, open ended connections to computing resources etc.

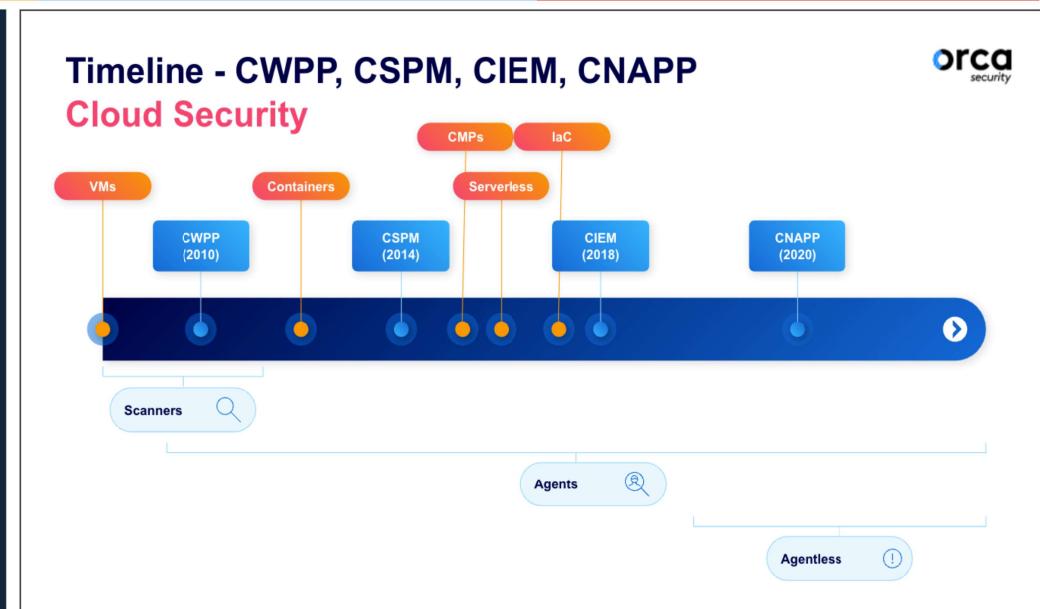
Cloud Native Application Protection Platform (CNAPP) : These are comprehensive security and compliance solutions specifically designed to safeguard cloud-native applications across the software development lifecycle from build to production. CNAPP promises an automated approach that will unify security capabilities that were previously siloed, reduce complexity, and put an end to inefficient solutions that can't scale.

CWPP	Supported Platforms	Key Features	Alert Generation	Machine learning	Free Trial	Pricing
AWS GuardDuty	AWS	Account-level threat detection security features, AWS services integration	Yes	Yes	Yes	\$1 per gigabyte per month
CloudGuard Workload Protection	AWS, Azure, GCP, VMware	Vulnerability management, network security, compliance management, threat intelligence, multi-cloud environments	Yes	Yes	Yes	Check Point has not provided pricing information for this product.
Illumio Core	AWS, Azure, GCP, VMware	Micro-segmentation, isolated visibility, policy enforcement, threat detection	Yes	Yes	Yes	\$7,000 per unit subscription per year.
Microsoft Defender for Cloud	Azure, AWS, Google Cloud Platform	Unified CWPP and CSPM security features across all platforms, advanced threat detection, Azure integrated compliance monitoring, vulnerability assessment	Yes	Yes	Yes	Starts at \$14.60 per server per month.
Orca Security	AWS, Azure, GCP	Cloud migration capabilities, vulnerability scanning, malware detection, data leak prevention	Yes	Yes	Yes	Orca has not provided pricing information for this product.
Prisma Cloud by Palo Alto	AWS, Azure, GCP	DevOps integration, Compliance monitoring, network security, container security, data loss prevention	Yes	Yes	Yes	Starts at \$9,000 annually per 100 Business Edition credits.
SentinelOne Singularity Cloud	AWS, Azure, GCP	Advanced automation capabilities, Asset discovery, vulnerability management, runtime protection, threat	Yes	Yes	Yes	Starts at \$36 per VM/kube node per month.

Security Issues in the Cloud - Remedy

CSPM vs CWPP
Which is the right choice for your business?

Parameters	CSPM	CWPP
Visibility	Cloud Architecture	Workload Inventory
Focus	Vulnerability Management	Endpoint Security
Checks	Configurations, Software, Libraries	Workloads, Applications, Containers
Dominant NIST Area	Identify & Protect	Detect and Respond
Access	API	API & Agent
Rights	Reader	Full
Threat Detection	Misconfigurations, Unauthorized Access, Compliance Violations, Data Exposure, Deployment Risks, Suspicious Network Traffic	Malware, Data Breaches, Unauthorized Access, Intrusions, DoS Attacks, Compliance Violations, Workload Vulnerabilities
Compliance Management	Overall Cloud Environment	Workload Specific
Protection	Alerts & Recommendations	Access Control & Encryption



Agent: Need installation of specific S/W on all Infra components.

Agentless: No S/W installed on each & every PC, but directed from a central location.

orca
security

Security Issues in the Cloud - Remedy

CSPM Vs CWPP: A Comparative Analysis (clouddefense.ai)

							Risk or Compliance Scoring	Automation	Advanced Data Governance and Security Capabilities	CIEM Capabilities	Pricing
CWPP	Supported Platforms	Key Features	Alert Generation	Machine learning	Free Trial	Pricing					
AWS GuardDuty	AWS	Account-level threat detection, security features, AWS services integration, vulnerability management, network security.	Yes	Yes	Yes	\$1 per gigabyte per month	Palo Alto Networks Prisma Cloud	Yes	Yes	Yes	Dependent upon selected partner and other factors; starting at \$18,000 for a 12-month 100 credit subscription through AWS Marketplace.
CloudGuard Workload Protection	AWS, Azure, GCP, VMware	Cloud workload protection, threat intelligence, multi-cloud environments, micro-segmentation, workload visibility, policy enforcement, threat detection.	Yes	Yes	Yes	Check Point has not provided pricing information for this product.	Check Point CloudGuard Cloud Security Posture Management	Yes	Yes	Yes	Dependent upon selected partner and other factors; starting at \$2625 per month for 25 assets through AWS Marketplace.
Illumio Core	AWS, Azure, GCP, VMware	Unified CWPP and CSPM security platform across all platforms, advanced threat detection.	Yes	Yes	Yes	\$7000 per unit subscription per year.	Lacework	Yes	Yes	Limited	Dependent upon selected partner and other factors; starting at \$1,500 per month plus \$0.01 usage fee per Enterprise Cloud Security Platform unit used.
Microsoft Defender for Cloud	Azure, AWS, Google Cloud Platform	Cloud configuration capabilities, threat visibility scanning, malware detection, data leak prevention, DevOps integration, threat detection monitoring, vulnerability assessment.	Yes	Yes	Yes	Starts at \$14.80 per server per month.	CrowdStrike Falcon Cloud Security	Yes	Yes	Yes	Dependent upon selected partner and other factors; starting at \$14.80 for a 12-month subscription through AWS Marketplace.
Orca Security	AWS, Azure, GCP	Cloud configuration capabilities, threat visibility scanning, malware detection, data leak prevention, DevOps integration, threat detection monitoring, vulnerability assessment.	Yes	Yes	Yes	Orca has not provided pricing information for this product.	CyberArk CyberArk Cloud Security	Yes	Yes	Yes	Dependent upon selected partner and other factors; Pro plan starts at \$10,000 for a 12-month subscription through Azure Marketplace.
Prisma Cloud by Palo Alto	AWS, Azure, GCP	Cloud workload protection, threat detection, monitoring, network security, container security, data loss prevention, automated automation capabilities, Asset discovery, vulnerability management, runtime protection, threat.	Yes	Yes	Yes	Starts at \$300 annually per 100 Business Edition credits.	Trend Micro Trend Cloud One Conformity	Yes	Yes	Limited	Limited
SentinelOne SingularCloud	AWS, Azure, GCP	Cloud workload protection, threat detection, monitoring, network security, container security, data loss prevention, automated automation capabilities, Asset discovery, vulnerability management, runtime protection, threat.	Yes	Yes	Yes	Starts at \$36 per vm/kubernetes worker node, per month.	Ermetic	Limited	Yes	Limited	Dependent upon selected partner and other factors; Commercial plan starts at \$28,000 for a 12-month subscription through AWS Marketplace.

Minimize Lack of Trust: Policy Language

Consumers have specific security needs but don't have a say-so in how they are handled

Currently consumers cannot dictate their requirements to the provider (SLAs are one-sided)

Standard language to convey one's policies and expectations

Agreed upon and upheld by both parties

Standard language for representing SLAs

Create policy language with the following characteristics:

Machine-understandable (or at least processable),

Easy to combine/merge and compare

BITS Pilani

Minimize Lack of Trust: Certification

Certification

Some form of reputable, independent, comparable assessment and description of security features and assurance

Sarbanes-Oxley, DIACAP, DISTCAP, etc

Risk assessment

Performed by certified third parties

Provides consumers with additional assurance

Minimize Loss of Control: Monitoring

Cloud consumer needs situational awareness for critical applications

When underlying components fail, what is the effect of the failure to the mission logic

What recovery measures can be taken by provider and consumer

Requires an application-specific run-time monitoring and management tool for the consumer

The cloud consumer and cloud provider have different views of the system

Enable both the provider and tenants to monitor the components in the cloud that are under their control

BITS Pilani

Minimize Loss of Control: Monitoring

Provide mechanisms that enable the provider to act on attacks he can handle.

infrastructure remapping

create new or move existing fault domains

shutting down offending components or targets and assisting tenants with porting if necessary Repairs

Provide mechanisms that enable the consumer to act on attacks that he can handle

application-level monitoring

RAdAC (Risk-adaptable Access Control)

VM porting with remote attestation of target physical host

Provide ability to move the user's application to another cloud

Minimize Loss of Control: Monitoring

The concept of 'Don't put all your eggs in one basket'

Consumer may use services from different clouds through an intra-cloud or multi-cloud architecture

A multi-cloud or intra-cloud architecture in which consumers

- Spread the risk

- Increase redundancy (per-task or per-application)

- Increase chance of mission completion for critical applications

Possible issues to consider:

- Policy incompatibility (combined, what is the overarching policy?)

- Data dependency between clouds

- Differing data semantics across clouds

- Knowing when to utilize the redundancy feature

 - monitoring technology

 - Is it worth it to spread your sensitive data across multiple clouds?

 - Redundancy could increase risk of exposure

BITS Pilani

Minimize Loss of Control: Access Control

Many possible layers of access control

E.g. access to the cloud, access to servers, access to services, access to databases (direct and queries via web services), access to VMs, and access to objects within a VM

Depending on the deployment model used, some of these will be controlled by the provider and others by the consumer

Regardless of deployment model, provider needs to manage the user authentication and access control procedures (to the cloud)

Federated Identity Management: access control management burden still lies with the provider

Requires user to place a large amount of trust on the provider in terms of security, management, and maintenance of access control policies.

This can be burdensome when numerous users from different organizations with different access control policies, are involved

Minimize Multi-tenancy

Can't really force the provider to accept less tenants

Can try to increase isolation between tenants

Strong isolation techniques (VPC to some degree)

QoS requirements need to be met

Policy specification

Can try to increase trust in the tenants

Who's the insider, where's the security boundary? Who can I trust?

Use SLAs to enforce trusted behavior

BITS Pilani

Conclusion

Cloud computing is sometimes viewed as a reincarnation of the classic mainframe client-server model

However, resources are ubiquitous, scalable, highly virtualized

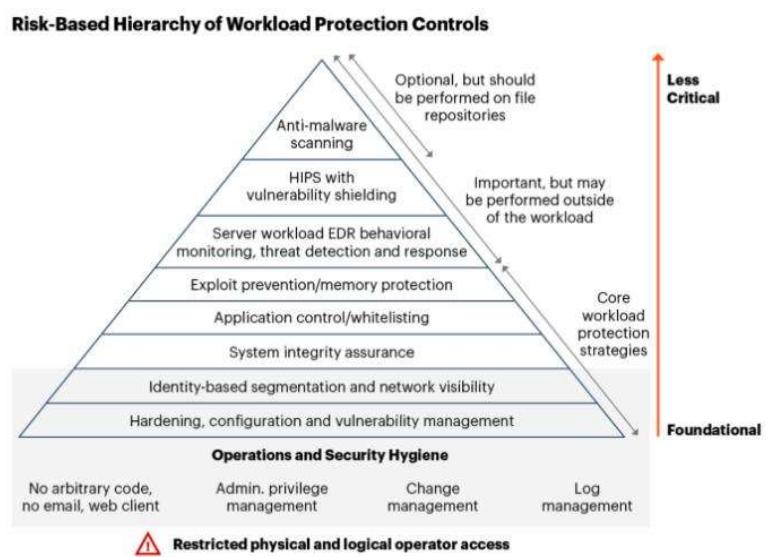
Contains all the traditional threats, as well as new ones

In developing solutions to cloud computing security issues it may be helpful to identify the problems and approaches in terms of

Loss of control

Lack of trust

Multi-tenancy problems



Source: Gartner

Gartner's risk based hierarchy of cloud workload protection controls

Agenda

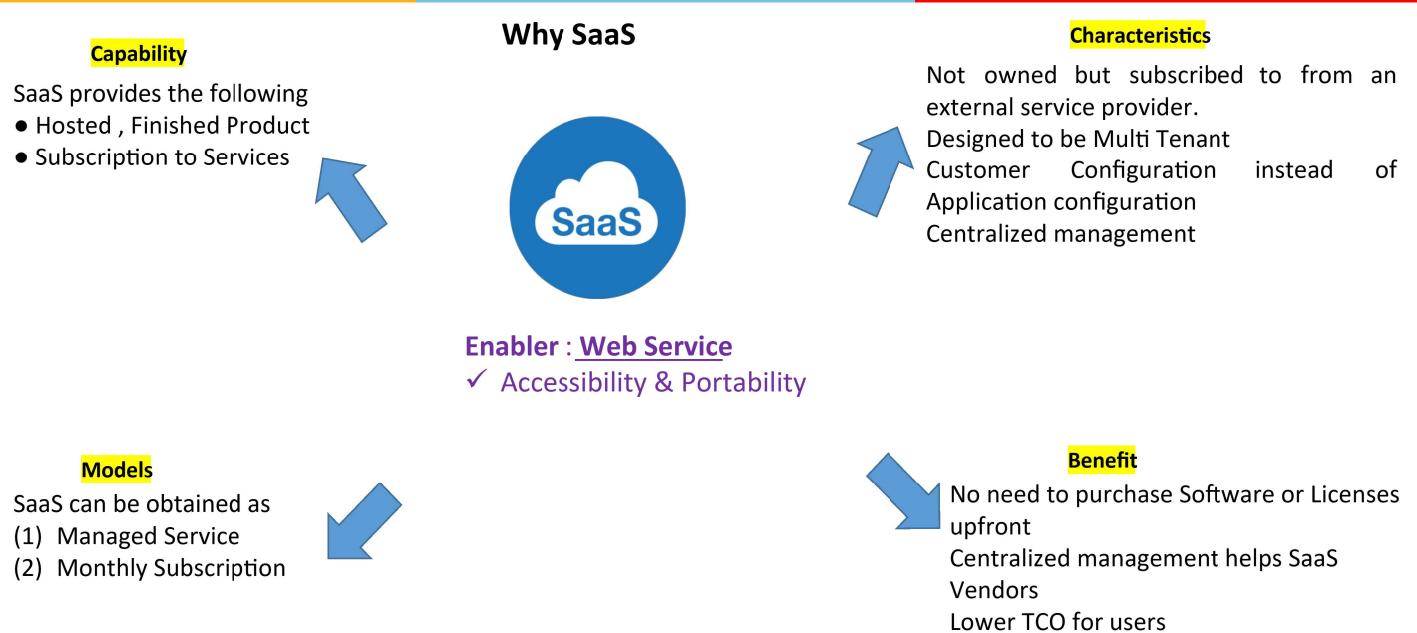


- ❖ SaaS Recap
- ❖ Capacity management in Cloud computing
- ❖ Virtual Infrastructure management
- ❖ RESERVIOR Project
- ❖ Distributed management of virtual machines
- ❖ Reservation-based provisioning of virtualized resource
- ❖ Provisioning to meet SLA commitments
- ❖ Scheduling models and algorithms
- ❖ Haieza project

2

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

Software as a Service - Summary



True SaaS Applicability



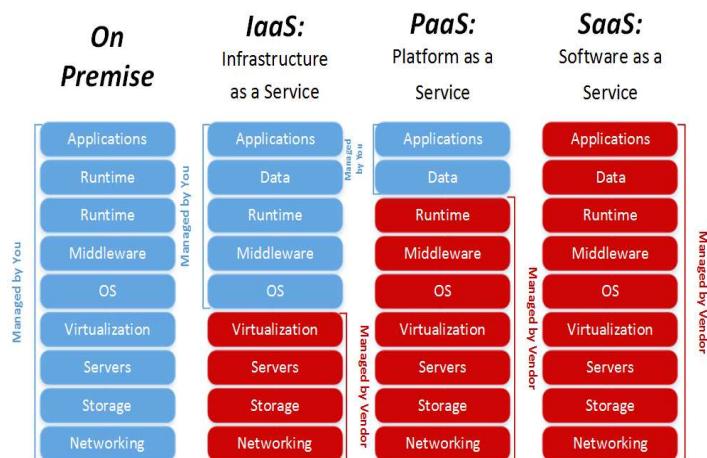
- **Enterprise Software Application**

- Perform business functions
- Organize internal and external information
- Share data among internal and external users
- The most standard type of software applicable to SaaS model
- Example: Salesforce.com CRM application, Siebel On-demand application

BITS Pilani

PARAMETER	SAAS	PAAS	IAAS
Full Form	Software As a Service	Platform as a Service	Infrastructure as a Service
General Users	Business Users	Developers and Deployers	System managers
Services Available	Email , Office automation , CRM , website testing , Virtual desktop	Service and application test , development , integration and deployment	Virtual machines, operating systems, network, storage, backup services.
Business Justification	To complete business tasks	Create and deploy service and applications for users	Create platform for service and application test, development.
Examples	Paypal , Salesforce.com	Azure Service platform, Force.com	Amazon EC2 , GoGrid
Control	Highest degree of control and flexibility	Good degree of control and flexibility	Minimal degree of control and flexibility
Operational Cost	Minimal	Lower	Highest
Portability	No portability	Lower	Best
Risk Of Vendor Interlock	Highest	Medium	Lowest
Security	Requires transparency in service provider's security policies to be able to determine the degree of sensitive corporate data.	Additional security is required to make sure rogue applications don't exploit vulnerabilities in software platform.	Should consider Virtual and physical servers security policy conformity.

SaaS vs PaaS vs IaaS



CRUCIAL
CLOUD HOSTING

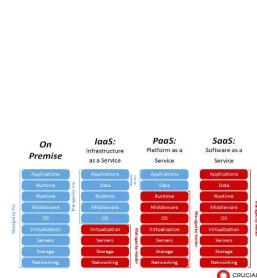
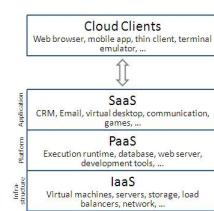
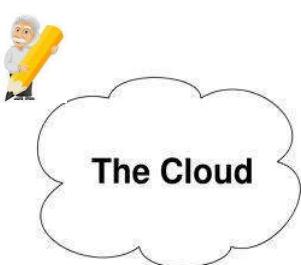
COMPARISON



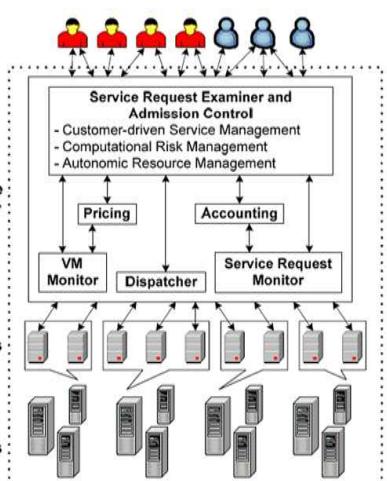
Capacity Management



Anatomy of a Cloud Ecosystem



Users/
Brokers



CLOUD requires managing..... But what will you manage?



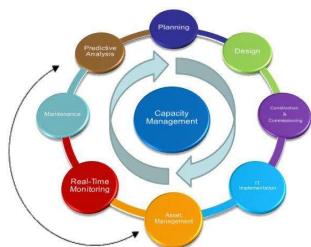
- Cloud Distributed environment

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

Why Capacity Management in Cloud?

Capacity Management is

- Strategic and Proactive
- Assists in managing service quality
- Assists in managing cost expenditures
- Aligns business and IT
- Match needs and cost during growth



Following are five characteristics of Cloud

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

BITS Pilani

What are the Challenges?



Capacity on Demand-
Too Little, Too Much
or Just Right?



Automation and
Control



Security Protecting
your loved ones"



Scalability- In / Up or
Out?



Old Habits – The
Potential Risks



Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.
- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

Importance of Capacity Management

When Capacity is Too Little (under capacity)

- Application Sizing not performed
- Controlled by limits
- Lower usage costs
- Service Levels affected – by how much?
- High impact on business?
 - Loss of service
 - SLA breach penalties



© Andrew Gray | Photosensitivity.com

When Capacity is Too Much (over capacity)

- Unlimited access to resources
- Service Levels unaffected
- Costs – higher resource usage, software licensing
- Impacts on other services
 - Poor performance
 - Wasted resources (multi virtual CPU (vSMP) & single-threaded applications)
 - VM Sprawl
 - Increased pressure to manage virtual resources



- Gartner – "most organizations overprovision their infrastructure by at least 100%"

BITS Pilani

So What's the Way? – Find the Right Balance

We need to have the Right Capacity?



- Application Sizing performed
- Efficient use of IaaS
- Acceptable Service Levels
 - continuous review and adjustment
- Controlled by shares, limits and reservations
- Continuous monitoring and tuning
 - Configuration adjustments (CPU, Memory)
 - VM consolidation
 - Power off unused ESX hosts
- Right Balance
 - Find the equilibrium (service / cost)



How To ? – Find the Right Balance



Automation and Control

- Rapid Elasticity- Guest Migration and Portability, Templates, Golden Host
- Resource Pools (limits, shares and reservations)
- Load Balancing
 - DRS (Automatic, Partial or Manual)
- Affinity
 - VM to VM or VM to Host
 - CPU
 - Fine grained tweaking for performance gains – Single Threaded Apps
 - Licensing

Protecting your loved ones

- Critical business applications
- Service Levels must be met
- Highest Priority (shares, unlimited), CPU affinity
- Must guarantee resources (reservations)
- High Availability Clusters
 - VMware - Fault Tolerance
- Trade offs
 - "just in case" capacity management could impact on other services
 - Significant impact? Think about scaling out or up .

BITS Pilani

How To ? – Find the Right Balance



Old Habits – Potential Risks

- Gartner – “Through 2015, more than 70% of private cloud implementations will fail to deliver operational, energy and environmental efficiencies.”
- Infrastructure or application hugging
 - Silo mentality “what’s mine is mine”
- Lack of resource sharing
 - Through lack of trust and confidence
- “Just in case” capacity planning
 - Leads to over provisioning



What is Virtual Infrastructure Management

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.



BITS Pilani

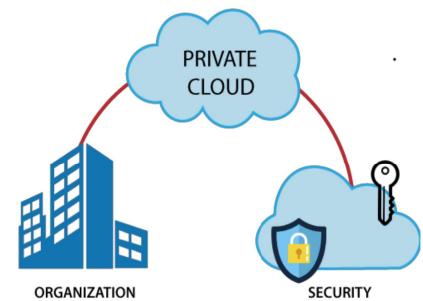
Virtual Infrastructure Management

- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.



Virtual Infrastructure Management

- Virtual infrastructure management in **private clouds** has to deal with an additional problem:
- **Unlike** large IaaS cloud providers such as Amazon, private clouds typically do not have enough resources to provide the illusion of “infinite capacity.”
- The immediate provisioning scheme used in public clouds, where resources are provisioned at the moment they are requested, is **ineffective in private clouds**.

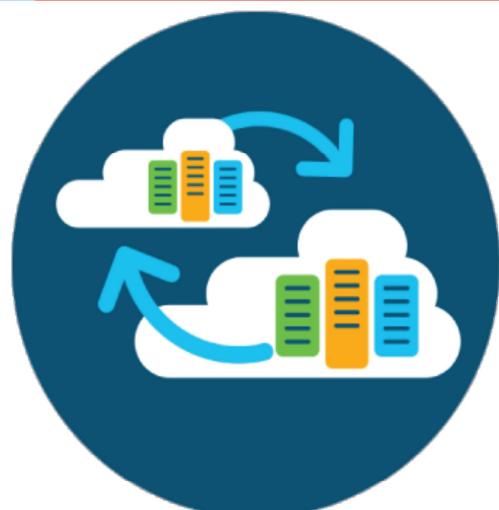


BITS Pilani

So What is the way out?

Managing virtual infrastructures in a private/hybrid cloud is a different problem than managing a virtualized data center, and existing tools lack several features that are required for building IaaS clouds.

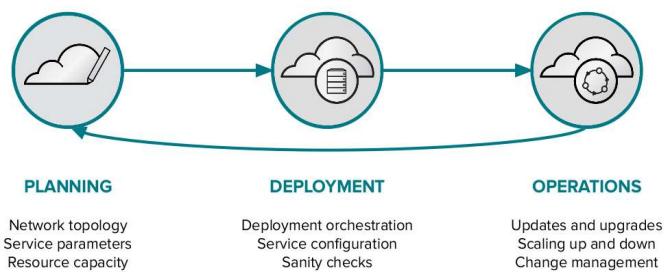
- Traditional methods can only operate with some preconfigured placement policies, which are generally simple (round robin, first fit, etc.) and based only on CPU speed and utilization of a fixed and predetermined number of resources, such as memory and network bandwidth.
- Thus, there are still several gaps in existing VI solutions. Filling these gaps will require addressing a number of research challenges such as virtual machine management, resource scheduling, SLAs, federation of resources, and security.



VIM



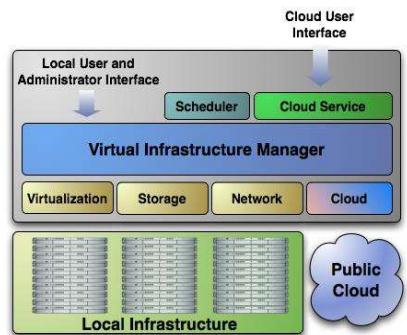
Virtual Infrastructure Manager (VIM)



What is a Virtual Infrastructure Manager?



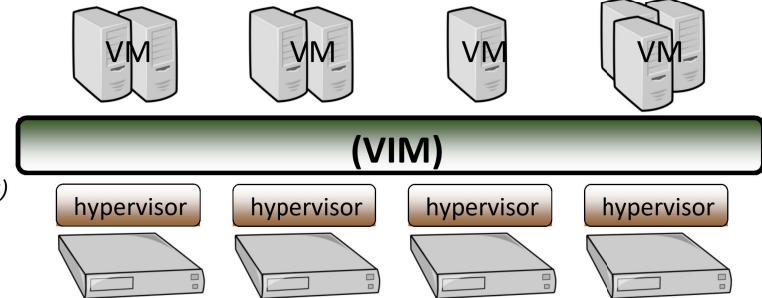
- A VIM runs on top of a hypervisor in a virtualized environment. The hypervisor allocates and manages virtual machines. The VIM deals with the allocation of resources in the virtual infrastructure. They include computational resources (processors), storage, and network resources. Virtual infrastructure management allows the allocation to happen based on current requirements, rather than being statically allocated.
- The VIM carries out several tasks:
- Allocating resources in accordance with traffic engineering rules.
- Support for defining operational rules.
- Definition of hub-to-facility mapping.
- Providing information for provisioning virtual infrastructure orchestration (VIO).



Why a Virtual Infrastructure Manager?

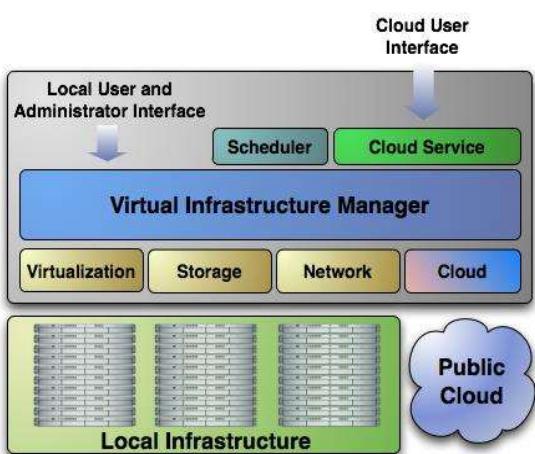


- VMs are great!!...but something more is needed
- Where did/do I put my VM? (*scheduling & monitoring*)
- How do I provision a new cluster node? (*clone*)
- What IP addresses are available? (*networking*)
- Provide a *uniform view* of the resource pool
- *Life-cycle management* and monitoring of VM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



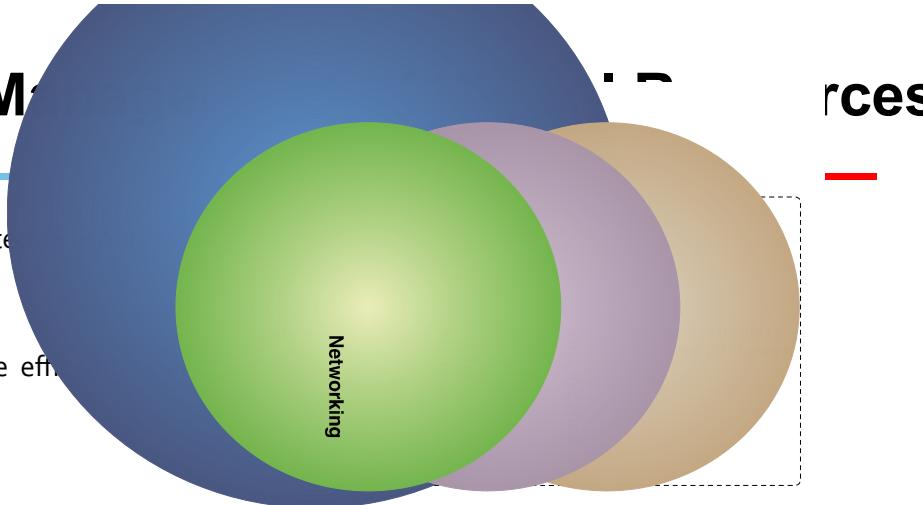
Why a Virtual Infrastructure Manager?

- 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



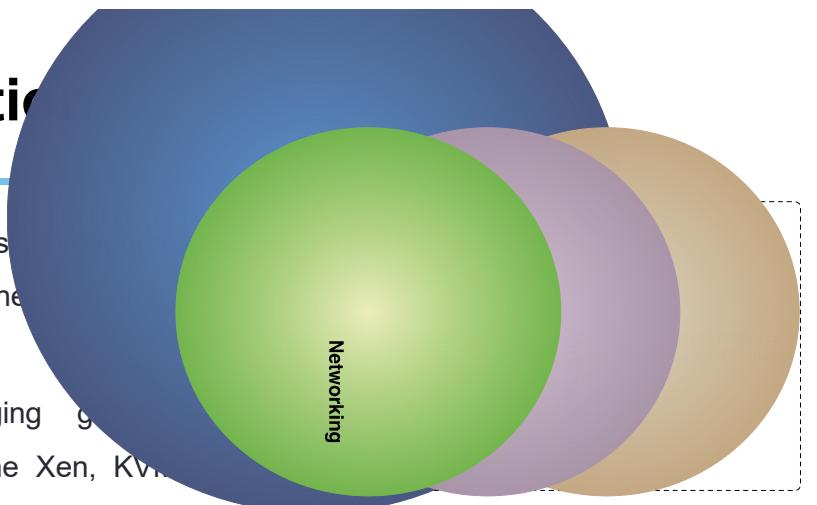
- Backend of Public Cloud: Internal management of the infrastructure
- Private Cloud: Virtualization of cluster or data-center for internal users
- Cloud Interoperation: On-demand access to public clouds

Enter – Distributed Management



- The cloud ecosystem is inherently distributed and requires resources to be spread around.
- Location also plays an important role in the efficient selection and placement of optimal selection & placement of resources.
- The problem of efficiently selecting or scheduling computational resources is well known. However, the state of the art in **VM-based resource scheduling** follows a **static approach**, where resources are initially selected using a greedy allocation strategy, with minimal or no support for other placement policies.
- To efficiently schedule resources, **VI managers** must be able to support **flexible and complex scheduling policies** and must *leverage* (use) the ability of VMs to suspend, resume, and migrate.

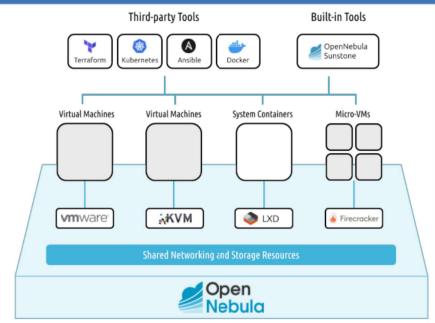
So What is the Solution?



- Managing VMs in a pool of distributed physical hosts is a key concern in IaaS clouds, requiring the services of a distributed infrastructure manager.
- **OpenNebula** is capable of managing geographically distributed interconnected VMs—with support for the Xen, KVM, and VMWare platforms—within data centers and private clouds that involve a large amount of virtual and physical servers.
- **OpenNebula** can also be used to build hybrid clouds by interfacing with remote cloud sites.

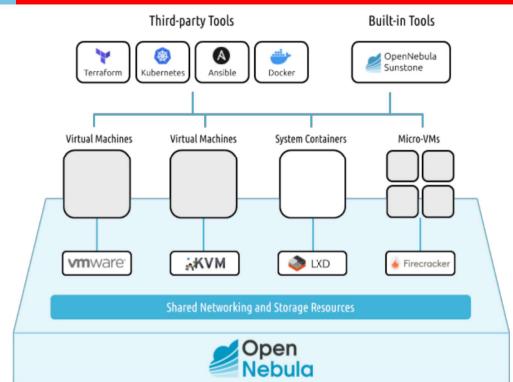


OpenNebula VIM



What is OpenNebula

- OpenNebula is a simple, feature-rich and flexible solution for the management of virtualized data centers.
- It enables private, public and hybrid clouds. Here are a few facts about this solution.
- OpenNebula is an open source cloud middleware solution that manages heterogeneous distributed data center infrastructures.
- It is designed to be a simple but feature-rich, production-ready, customizable solution to build and manage enterprise clouds—simple to install, update and operate by the administrators; and simple to use by end users.
- OpenNebula combines existing virtualization technologies with advanced features for multi-tenancy, automated provisioning and elasticity.
- A built-in virtual network manager maps virtual networks to physical networks.
- Distributions such as Ubuntu and Red Hat Enterprise Linux have already integrated OpenNebula.
- OpenNebula supports Xen, KVM and VMware hypervisors.



- **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

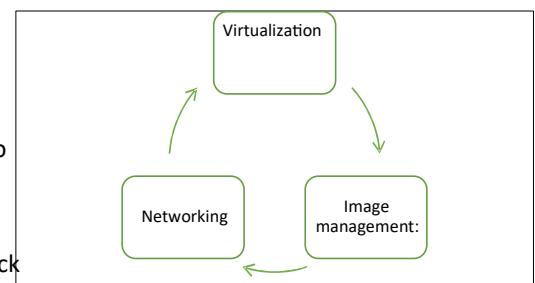
Stages of VM Life Cycle in OpenNebula

The life cycle of a VM within OpenNebula follows several stages:

- **Resource Selection:** allowing site administrators to configure the scheduler to prioritize the resources that are more suitable for the VM
- **Resource Preparation:** The disk images of the VM are transferred to the target physical resource. During the boot process, the VM is contextualized, a process where the disk images are specialized to work in a given environment.
- **VM Creation:** The VM is booted by the resource hypervisor
- **VM Migration:** The VM potentially gets migrated to a more suitable resource(e.g.,to optimizethe power consumption ofthe physical resources)
- **VM Termination:** When the VM is going to shut down, OpenNebula can transfer back its disk images to a known location. This way, changes in the VM can be kept for a future use.

Within OpenNebula, a VM is modeled as having the following attributes:

- A capacity in terms of memory and CPU
- A set of NICs attached to one or more virtual networks
- A set of disk images
- A state file (optional) or recovery file



VM Management in OpenNebula

- OpenNebula manages VMs by interfacing with a physical resource's hypervisor, such as Xen, KVM, or VMWare,Hyper-V to control (e.g., boot, stop, or shutdown) the VM;
- using a set of **pluggable drivers** that decouple the managing process from the underlying technology.
- Thus, whenever the core needs to manage a VM, it uses **high-level commands** such as “start VM,” “stop VM,” and so on, which are translated by the drivers into commands that the virtual machine manager can understand. By decoupling the OpenNebula core from the virtualization technologies through the use of a **driver-based architecture**, adding support for additional virtual machine managers only requires writing a driver for it.

What Does a Virtual Machine Manager Do?

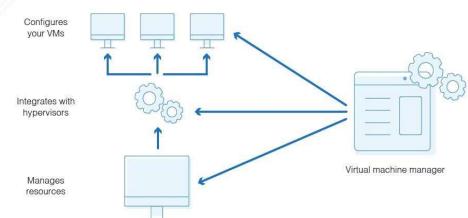
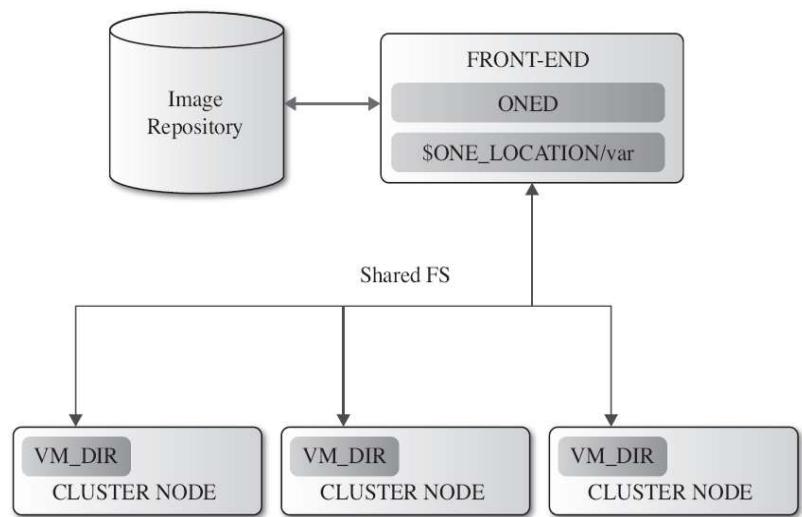


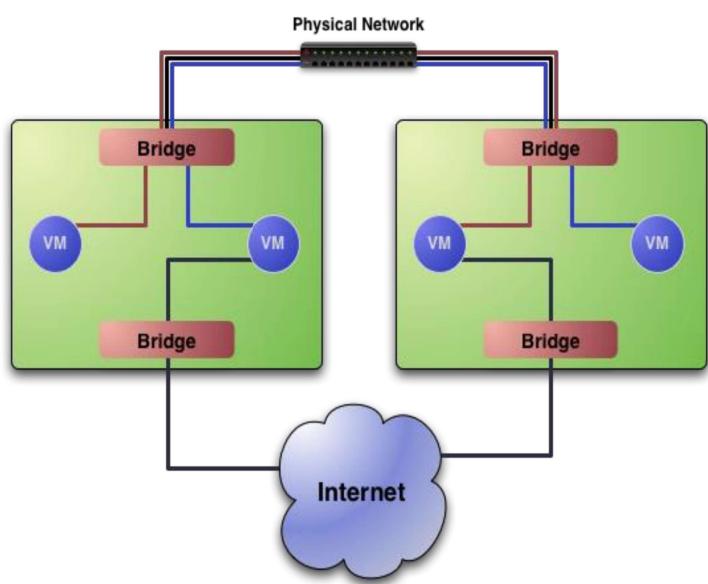
Image Management in OpenNebula

- Transferring the VM images from an image repository to the selected resource and by creating on-the-fly temporary images
- What is image?
 - Virtual disk contains the OS and other additional software
- Image management model



Networking OpenNebula

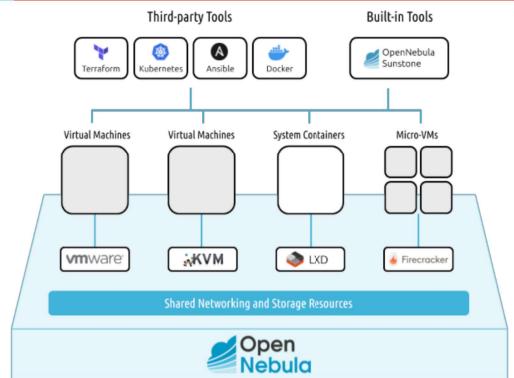
- In general, services deployed on a cloud require several interrelated VMs, with a virtual application network (VAN) being the primary link between them.
- OpenNebula dynamically creates these VANs and tracks the MAC addresses leased in the network to the service VMs.
- **physical cluster** as a set of hosts with one or more network interfaces, each of them connected to a different physical network.



Benefits of OpenNebula

For the Infrastructure Manager

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



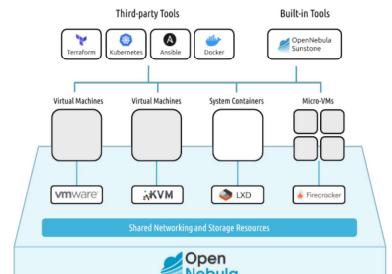
For the Infrastructure User

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

BITS Pilani

Features of OpenNebula

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



Comparison with Similar Technologies

	Platform ISF	VMware Vsphere	Eucalyptus	Nimbus	OpenNebula
Virtualization Management	VMware, Xen	VMware	Xen, KVM	Xen	Xen, KVM, VMware
Virtual Network Management	Yes	Yes	No	Yes	Yes
Image Management	Yes	Yes	Yes	Yes	Yes
Service Contextualization	No	No	No	Yes	Yes
Scheduling	Yes	Yes	No	No	Yes
Administration Interface	Yes	Yes	No	No	Yes
Hybrid Cloud Computing	No	No	No	No	Yes
Cloud Interfaces	No	vCloud	EC2	WSRF, EC2	EC2 Query, OGF OCCI
Flexibility and Extensibility	Yes	No	Yes	Yes	Yes
Open Source	No	No	GPL	Apache	Apache

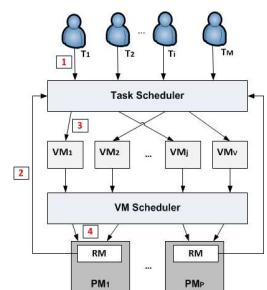
BITS Pilani



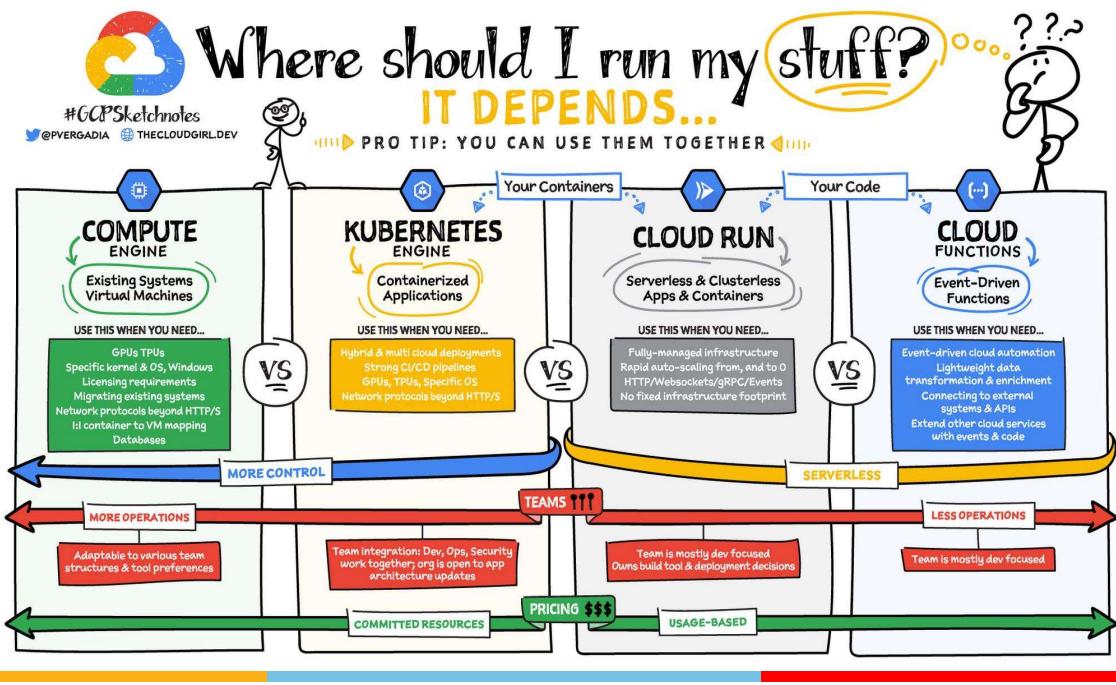
BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Scheduling VM Workloads



What is a Cloud Workload



BITS Pilani

What is a Cloud Workload

- A cloud workload is a specific application, service, capability or a specific amount of work that can be run on a cloud resource. Virtual machines, databases, containers, Hadoop nodes and applications are all considered cloud workloads.
- A workload is a **collection of resources and code that delivers business value, such as a customer-facing application or a backend process**. A workload might consist of a subset of resources in a single cloud account or be a collection of multiple resources spanning multiple cloud accounts.
- Examples of workloads are **marketing websites, e-commerce websites, the back-ends for a mobile app, analytic platforms, etc.** Workloads vary in levels of architectural complexity, from static websites to architectures with multiple data stores and many components.

Seven workloads your customers should move to cloud now

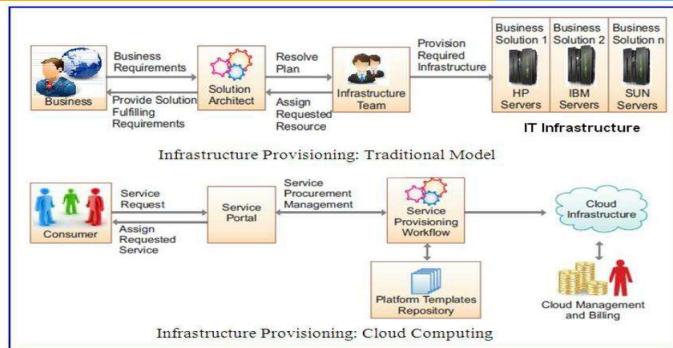


gartner.com/SmarterWithGartner

Source: Gartner
© 2020 Gartner, Inc. All rights reserved. Legal_1113170

Gartner

What are we Talking about



Top 5 Workloads Per Cloud Provider

Microsoft Azure	Amazon AWS	Google GCP
Database	Web/Content Hosting	Database
Industry Market Solutions	Analytics	Software Development/DevOps
Web/Content Hosting	Database	Industry Market Solutions
IoT/Data Streaming	AI/Cognitive/Machine Learning	Analytics
Software Development/DevOps	IoT/Data Streaming	Legacy/App Migration to Containers

Workload → Amount of work (or load) that software imposes on the underlying computing resources.

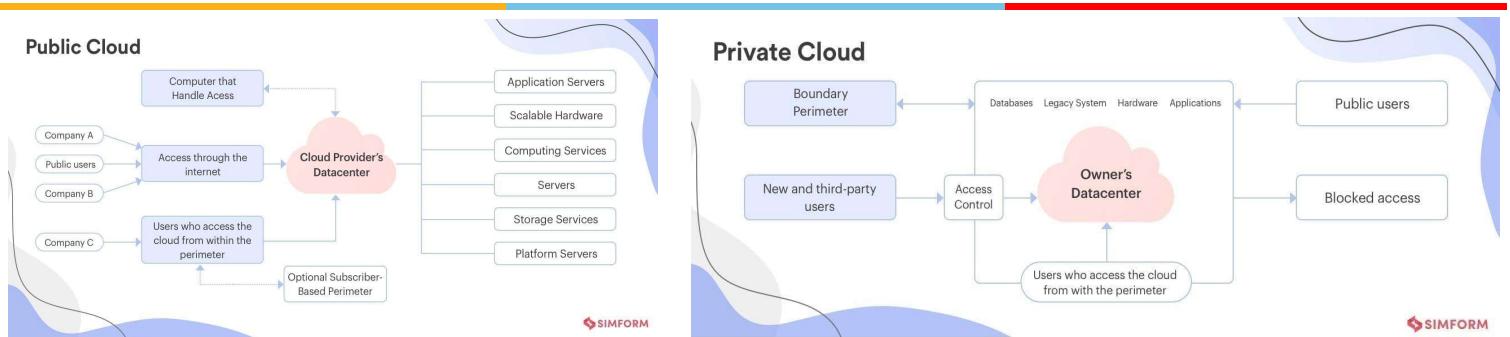
Workload → Amount of time and computing resources required to perform a specific task or produce an output from inputs provided.

Types of Workload → Static (fixed work always) Dynamic (Ad-hoc requests) Analytical (Big Data) Transactional (Main Frame)

Standardized metrics used to measure and report on an application's performance or load are collectively referred to as *benchmarks*.

BITS Pilani

Where do we Run Workloads?



Workload deployment -- determining where and how the workload runs -- is an essential part of workload management. Today, an enterprise can choose to deploy a workload on premises, as well as to a cloud.

Traditionally, workloads are deployed in the **enterprise data center**, which contains all of the **server, storage, network, services and other infrastructure required to operate the workload**. The business owns the data center facility and computing resources and fully controls the provisioning, optimization, and maintenance of those resources. The enterprise establishes policies and practices for the data center and workload deployment in order to meet prevailing **business goals and regulatory obligations**.

With the rise of the internet, cloud computing is now a viable alternative for many on-premises workload deployments.

The **challenge for any business is deciding just where to deploy a given workload**. Today, most general-purpose workloads can operate successfully in the public cloud, and, increasingly, applications are designed and developed to run natively and solely in a public cloud.

Workload Challenges

Technologically, the **most demanding workloads** may struggle in the **public cloud**. Some workloads require high-performance network storage or depend on internet throughput.

For example, database clusters that need high throughput and low latency may be unsuited to the cloud -- and the cloud provider may offer high-performance database services as an alternative. Applications that rely on low latency or are not designed for distributed computing infrastructures are usually kept on premises.

Technical issues aside, a business may decide to **keep workloads on-premises** for business continuance or regulatory reasons.

Cloud clients have **little actual insight into the underlying hardware** and other infrastructure that hosts the workloads and data. That can be problematic for businesses obligated to meet data security and other regulatory requirements such as clear auditing and proof of data residency. Keeping those sensitive workloads in the local data center allows the business to control its own infrastructure and implement the necessary auditing and controls.

BITS Pilani

Workload Challenges

Cloud providers are also **independent businesses** that serve their own business interests and may not be able to meet an enterprise's specific **uptime or resilience expectations for a workload**. Outages happen and may last for hours -- even days -- adversely affecting client businesses and their customer base. Consequently, organizations often opt to keep critical workloads in the local data center where dedicated IT staff can maintain them.

Some organizations implement a **hybrid cloud strategy** that mixes on-premises, private cloud and public cloud services. This provides flexibility to run workloads and manage data where it makes the most sense, for reasons ranging from costs to security to governance and compliance. This presents tradeoffs -- for example, an organization may keep sensitive data and workloads in its own data center to preserve more direct control over them, but it also takes on more security responsibilities for them.

Cloud Workload – An Anatomy

Key Terms to Understand

Lease: A lease is defined as a contract between the Cloud Service Provider and the end user to facilitate the usage of resources available with the CSP to execute a workload of the end user.

Application: One or more business process encapsulated in code which requires computing resources to execute and provide business value.

Job: A Work Breakdown Structure of an application. An application may contain several jobs, and all of them need to be executed to complete execution of the application.

Tasks: Steps that need to be performed to complete a job. Task can be sequential or parallel.

When we talk about cloud workload we are referring to the completion of a specific job which provides some business values.

Resource Allocation vs Task Scheduling

- *It might look similar, but there are lots of differences*
- *RA → Identifying and allocating a resource of a type of work load*
- *TS → Ability to shuffle and manage tasks to operate efficiently on the available resources.*

BITS Pilani

Amdahl's Law Formula

$$S_{max} = \frac{1}{(1-p) + \frac{p}{s}}$$

In computer programming, Amdahl's law is that, in a program with parallel processing, a relatively few instructions that have to be performed in sequence will have a limiting factor on program speedup such that adding more processors may not make the program run faster.

This is generally an argument against parallel processing for certain applications and, in general, against overstated claims for parallel computing. Others argue that the kinds of applications for which parallel processing is best suited tend to be larger problems in which scaling up the number of processors does indeed bring a corresponding improvement in throughput and performance.

Amdahl's law formula calculates the expected speedup of the system if one part is improved. It has three parts: S_{max} , p , and s .

S_{max} is the maximum possible improvement of the overall system. It is expressed as a decimal greater than 1. If the operation is improved to be done in half the time, $S_{max} = 2$. Higher means a greater improvement.

p is the part of the system to be improved, expressed as a number between 0-1. If the part is 45% of the system, $p = 0.45$.

s is the improvement factor of p , expressed by how many times faster p can be done. If it can be done in 1/3rd the time, then $s = 3$.

Essentially, the equation subtracts out the part to be improved, then puts it back in after it has been improved.

Scheduling Techniques

- While a VI manager like OpenNebula can handle all the minutiae of managing VMs in a pool of physical resources, **scheduling these VMs efficiently is a different and complex matter.**
- **Immediate provisioning model** is used by commercial cloud providers, such as Amazon, since their data center capacity is assumed to be infinite.
- Best-effort provisioning where requests have to be queued and prioritized
- **Advance provisioning** where resources are pre-reserved so they will be guaranteed to be available at a given time period.
- However, when managing a private cloud with limited resources, an immediate provisioning model is insufficient.
- **A lease-based resource provisioning model** that can act as a scheduling back-end for OpenNebula, supporting other provisioning models other than the immediate provisioning models in existing cloud providers. In particular, Haizea adds support for both best-effort provisioning and advance reservations, when managing a finite number of resources.

BITS Pilani

Scheduling Techniques – Existing Approaches

- Efficient reservation of resources in resource management systems has been studied considerably, particularly in the context of job scheduling.
- In fact, most modern job schedulers support **advance reservation of resources**, but their implementation falls short in several aspects.
- First of all, they are constrained by the **job abstraction**; when a user makes an advance reservation in a job-based system, the user does not have direct and unfettered access to the resources. Cloud users can access the VMs they requested, and are allowed to submit jobs to them.
- Example-1: XYZ Inc creates a new queue that will be bound to the reserved resources, guaranteeing that jobs submitted to that queue will be executed on them (assuming they have permission to do so)
- Example-2: ABC Inc, simply allow users to specify that a submitted job should use the reserved resources (if submitting user has permission to do so).

Scheduling Techniques – Existing Approaches

- Additionally, advance reservations lead to utilization problems caused by the need to vacate resources before a reservation can begin.
- Traditional job schedulers are unable to efficiently schedule workloads combining both best-effort jobs and advance reservations.
- However, advance reservations can be supported more efficiently by using a scheduler capable of preempting running jobs at the start of the reservation and resuming them at the end of the reservation.
- Preemption can also be used to run large parallel jobs (which tend to have long queue times) earlier, and it is especially relevant in the context of urgent computing, where resources have to be provisioned on very short notice and the likelihood of having jobs already assigned to resources is higher.
- While preemption can be accomplished by canceling a running job, the least disruptive form of preemption is checkpointing, where the preempted job's entire state is saved to disk, allowing it to resume its work from the last checkpoint.

Scheduling Techniques – Existing Approaches

- Additionally, some schedulers also support job migration, allowing check-pointed jobs to restart on other available resources, instead of having to wait until the preempting job or reservation has completed.
- Check-pointing-based preemption, requires the job's executable itself to be checkpointable. An application can be made checkpointable by explicitly adding that functionality to an application (application-level and library-level checkpointing) OR transparently by using OS-level checkpointing, where the operating system (such as Cray, IRIX, and patched versions of Linux using BLCR [17]) checkpoints a process, without rewriting the program or relinking it with checkpointing libraries.
- Thus, a job scheduler capable of checkpointing-based preemption and migration could be used to checkpoint jobs before the start of an advance reservation, minimizing their impact on the schedule.
- However, the application and library-level checkpointing approaches burden the user with having to modify their applications to make them checkpointable, imposing a restriction on the software environment. On the other hand, OS-level checkpointing is a more appealing option, but still imposes certain software restrictions on resource consumers.

Scheduling Techniques – Existing Approaches

- An alternative approach to supporting advance reservations was proposed by Nurmi et al. [18], which introduced “virtual advance reservations for queues” (VARQ).
- This approach overlays advance reservations over traditional job schedulers by first predicting the time a job would spend waiting in a scheduler’s queue and then submitting a job (representing the advance reservation) at a time such that, based on the wait time prediction, the probability that it will be running at the start of the reservation is maximized.
- Since no actual reservations can be done, VARQ jobs can run on traditional job schedulers, which will distinguish between the regular best-effort jobs and the VARQ jobs.
- Although this is an interesting approach that can be realistically implemented in practice (since it does not require modifications to existing scheduler), it still depends on the job abstraction.

BITS Pilani

Scheduling Techniques – VM Overheads

Virtualization technologies are a key enabler of many features found in IaaS clouds. Virtual machines are also an appealing vehicle for implementing efficient reservation of resources due to:

- Ability to be suspended,
- Potentially migrated,
- Resumed without modifying any of the applications running inside the VM.

However, virtual machines also raise additional challenges related to the overhead of using VMs:

- **Preparation Overhead.** When using VMs to implement reservations, a VM disk image must be either prepared on-the-fly or transferred to the physical node where it is needed. Since a VM disk image can have a size in the order of gigabytes, this preparation overhead can significantly delay the starting time of leases. This delay may, in some cases, be unacceptable for advance reservations that must start at a specific time.
- **Runtime Overhead.** Once a VM is running, scheduling primitives such as **checkpointing** and **resuming** can incur significant overhead since a VM’s entire memory space must be saved to disk, and then read from disk. Migration involves transferring this saved memory along with the VM disk image. Similar to deployment overhead, this overhead can result in noticeable delays.

Reservation Based Provisioning

- A particularly interesting problem when provisioning virtual infrastructures is **how to deal with situations where the demand for resources is known beforehand**—for example, when an experiment depending on some complex piece of equipment is going to run from 2 pm to 4 pm, and computational resources must be available at exactly that time to process the data produced by the equipment.
- Commercial clouds do have **infinite resources** to handle this situation. On the other hand, when dealing with **finite capacity**, a different approach is needed. However, the intuitively simple solution of reserving the resources beforehand is not so simple, because it is known to cause resources to be underutilized, due to the difficulty of scheduling other requests around an inflexible reservation.

Provisioning to meet SLA

- IaaS clouds can be used to deploy services that will be consumed by users other than the one that has deployed the services.
- There is a distinction between the **cloud consumer** (i.e., the **service owner**; for instance, the company that develops and manages the applications) and the **end users** of the resources provisioned on the cloud (i.e., the **service user**; for instance, the users that access the applications).
- Furthermore, **service owners** will enter into **service-level agreements (SLAs)** with their end users, covering guarantees such as the timeliness with which these services will respond.
- Requirements are formalized in infrastructure SLAs between the **service owner** and **cloud provider**, separate from the high-level **SLAs between the service owner and its end users**.

Provisioning to meet SLA

- In many cases, either the service owner is not resourceful enough to perform an exact service sizing or service workloads are hard to anticipate in advance.
- Therefore, to protect high-level SLAs, the cloud provider should cater for **elasticity on demand**.
- **Scaling and de-scaling** of an application is best managed by the application itself. The reason is that in many cases, resource allocation decisions are **application-specific** and are being driven by the **application level metrics**.

Scheduling Techniques for Advance Reservation of Capacity

- Virtualization technologies are a key enabler of many features found in IaaS clouds. Virtual machines are also an appealing vehicle for implementing efficient reservation of resources due to:
 - Ability to be suspended,
 - Potentially migrated,
 - Resumed without modifying any of the applications running inside the VM.
- However, virtual machines also raise additional challenges related to the overhead of using VMs:
 - **Preparation Overhead.** When using VMs to implement reservations, a VM disk image must be either prepared on-the-fly or transferred to the physical node where it is needed. Since a VM disk image can have a size in the order of gigabytes, this preparation overhead can significantly delay the starting time of leases. This delay may, in some cases, be unacceptable for advance reservations that must start at a specific time.
 - **Runtime Overhead.** Once a VM is running, scheduling primitives such as **checkpointing** and **resuming** can incur in significant overhead since a VM's entire memory space must be saved to disk, and then read from disk. Migration involves transferring this saved memory along with the VM disk image. Similar to deployment overhead, this overhead can result in noticeable delays.

Solution – Haizea Scheduler

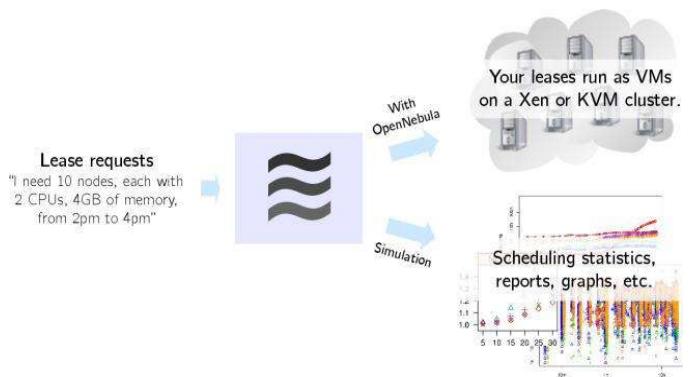
The Haizea project (<http://haizea.cs.uchicago.edu/>) was created to develop a scheduler that can efficiently support advance reservations efficiently by using the suspend/resume/migrate capability of VMs, but minimizing the overhead of using VMs.

The fundamental resource provisioning abstraction in Haizea is the lease, with three types of lease currently supported:

Advanced reservation leases, where the resources must be available at a specific time.

Best-effort leases, where resources are provisioned as soon as possible and requests are placed on a queue if necessary.

Immediate leases, where resources are provisioned when requested or not at all.



BITS Pilani

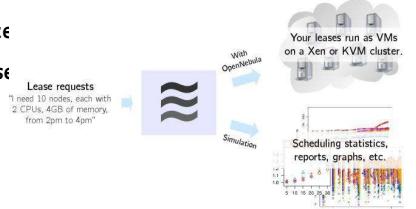
What is Haizea Scheduler

When managing a private cloud with limited resources, **an immediate provisioning model is insufficient**. A **lease-based resource provisioning model** that can act as a scheduling back-end for OpenNebula, **supporting other provisioning models other than the immediate provisioning models in existing cloud providers**. In particular, Haizea adds support for both best-effort provisioning and advance reservations, **when managing a finite number of resources**. The remainder of this section describes Haizea's leasing model and the algorithms Haizea uses to schedule these leases.

- We define a **lease** as “**a negotiated and renegotiable agreement between a resource provider and a resource consumer, where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumer.**”

- The terms must encompass the following:

- the hardware resources required by the resource consumer, such as CPUs, memory, and network bandwidth;
- a software environment required on the leased resources;
- and an availability period during which a user requests that the hardware and software resources be available.



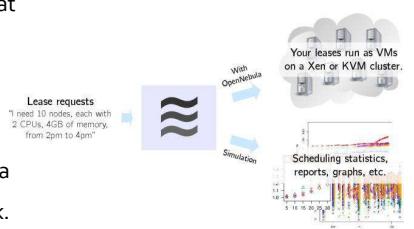
How Haizea Scheduler Works

We focus on the availability dimension of a lease and, in particular, on how to efficiently support advance reservations. Thus, we consider the following availability terms:

- Start time may be unspecified (a best-effort lease) or specified (an advance reservation lease). In the latter case, the user may specify either a specific start time or a time period during which the lease start may occur.
- Maximum duration refers to the total maximum amount of time that the leased resources will be available.
- Leases can be preemptable. A preemptable lease can be safely paused without disrupting the computation that takes place inside the lease.

Haizea's resource model considers that it manages W physical nodes capable of running virtual machines. Each node i has CPUs, megabytes (MB) of memory, and MB of local disk storage.

- We assume that all disk images required to run virtual machines are available in a repository from which they can be transferred to nodes as needed and that all are connected at a bandwidth of B MB/sec by a switched network.
- A lease is implemented as a set of N VMs, each allocated resources described by a *tuple* (p, m, d, b) , where p is number of CPUs, m is memory in MB, d is disk space in MB, and b is network bandwidth in MB/sec.
- A disk image I with a size of size (I) MB must be transferred from the repository to a node before the VM can start. When transferring a disk image to multiple nodes, we use multicasting and model the transfer time as size $(I)/B$.

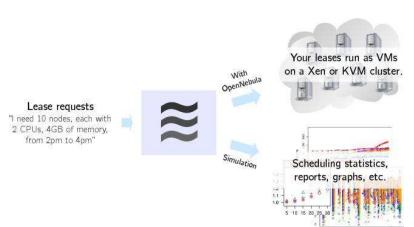


BITS Pilani

How Haizea Scheduler Works

Haizea is designed to process lease requests and determine how those requests can be mapped to virtual machines, leveraging their suspend/resume/migrate capability, in such a way that the leases' requirements are satisfied.

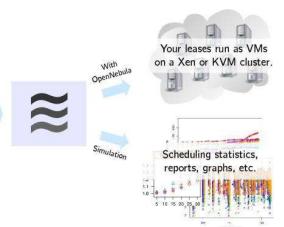
- The scheduling component of Haizea allows best-effort leases to be preempted if resources have to be freed up for advance reservation requests.
- Additionally, to address the preparation and runtime overheads mentioned earlier, the scheduler allocates resources explicitly for the overhead activities (such as transferring disk images or suspending VMs) instead of assuming they should be deducted from the lease's allocation.
- Besides guaranteeing that certain operations complete on time (e.g., an image transfer before the start of a lease), the scheduler also attempts to minimize this overhead whenever possible, most notably by reusing disk image transfers and caching disk images on the physical nodes.



How Haizea Scheduler Works

Best-effort leases are scheduled using a queue. When a best-effort lease is requested, the lease request is placed at the end of the queue, which is periodically evaluated using a backfilling algorithm to determine if any leases can be scheduled.

- The scheduler does this by first checking the earliest possible starting time for the lease on each physical node, which will depend on the required disk images. For example, if some physical nodes have cached the required disk image, it will be possible to start the lease earlier on those nodes.
- Once these earliest starting times have been determined, the scheduler chooses the nodes that allow the lease to start the soonest.
- The use of VM suspension/resumption allows the best-effort leases to be scheduled even if there are not enough resources available for their full requested duration.

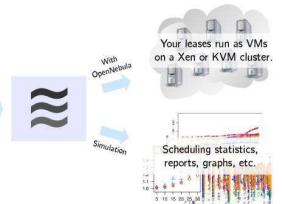


BITS Pilani

How Haizea Scheduler Works

Advance reservations, on the other hand, do not go through a queue, since they must start at either the requested time or not at all.

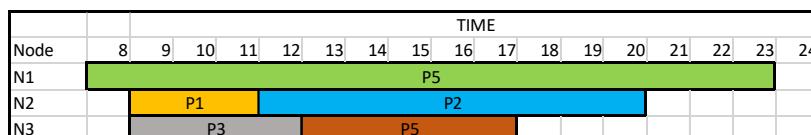
- Thus, scheduling this type of lease is relatively simple, because it mostly involves checking if there are enough resources available during the requested interval.
- However, the scheduler must also check if any associated overheads can be scheduled in such a way that the lease can still start on time.
- For preparation overhead, the scheduler determines if the required images can be transferred on time.
- These transfers are scheduled using an Earliest Deadline First (EDF) algorithm, where the deadline for the image transfer is the start time of the advance reservation lease.
- For runtime overhead, the scheduler will attempt to schedule the lease without having to preempt other leases; preemption is unavoidable. The necessary suspension operations are scheduled; if they can be performed on time.



Leasing Schedule – Best Effort Lease (BEL)

- Best-effort leases are scheduled using a **queue**. When a best-effort lease is requested, the lease request is placed at the end of the queue, which is periodically evaluated using a backfilling algorithm to determine if any leases can be scheduled.
- The scheduler does this by **first checking the earliest possible starting time** for the lease on each physical node, which will depend on the required disk images. **For example**, if some physical nodes have cached the required disk image, it will be possible to start the lease earlier on those nodes.
- Once these earliest starting times have been determined, the scheduler chooses the nodes that allow the lease to start the soonest.
- The use of **VM suspension/resumption** allows the best-effort leases to be scheduled **even if there are not enough resources available for their full requested duration**.

BITS Pilani



e.g. If we have three nodes N1, N2 and N3 and five processes have to be scheduled with entry time and duration of resources as follows – Best effort queue

Process	Entry time	Time required	Node	Start time		
P1	9am	3 hours	N2	9am	Ended at 12pm	
P2	10am	10 hours	N2	12pm	Ended at 8pm	
P3	9.30am	4 hours	N3	9.30am	Ended at 1.30pm	
P4	11am	5hours	N3	1.30pm	Ended at 6.30pm	
P5	8am	15hours	N1	8am	Ended at 11pm	

Leasing Schedule – Advanced Reservation (AR)

- Advance reservations, on the other hand, do not go through a queue, since they must start at either the requested time or not at all.
- Thus, scheduling this type of lease is relatively simple, because it mostly involves checking if there are enough resources available during the requested interval.
- However, the scheduler must also check if any associated overheads can be scheduled in such a way that the lease can still start on time.
- For preparation overhead, the scheduler determines if the required images can be transferred on time.
- These transfers are scheduled using an Earliest Deadline First (EDF) algorithm, where the deadline for the image transfer is the start time of the advance reservation lease.
- For runtime overhead, the scheduler will attempt to schedule the lease without having to preempt other leases; if preemption is unavoidable. The necessary suspension operations are scheduled; if they can be performed on time.

e.g. If we have three nodes N1, N2 and N3 and five processes have to be scheduled with start time and duration with P2 and P5 being advanced reservation

Process	Entry time	Time required	Node	Start time		
P1	9am	3 hours				
P2	10am	10 hours				
P3	9.30am	4 hours				
P4	11am	5hours				
P5	8am	15hours				

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- If temporal behavior of services with respect to resource demands is highly predictable, then capacity can be efficiently scheduled using reservations.
- In this section we focus on **less predictable elastic workloads**. For these workloads, **exact scheduling of capacity may not be possible**. Rather than that, **capacity planning and optimizations are required**.
- IaaS providers perform two complementary management tasks:
 - (1) **Capacity planning** to make sure that SLA obligations are met as contracted with the service providers and;
 - (2) **Continuous optimization** of resource utilization in specific workload to make the most efficient use of the existing capacity.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- Infrastructure SLA's

- IaaS can be regarded as a giant virtual hardware store, where computational resources such as virtual machines (VM), virtual application networks (VAN) and virtual disks (VD) can be **ordered on demand in the matter of minutes or even seconds**.
- Chandra et al. [29] quantitatively study advantages of **fine-grain resource allocation** in a shared hosting platform. As this research suggests, **fine-grain temporal and spatial resource allocation** may lead to substantial improvements in capacity utilization.
- Amazon EC2 [1] offers small, large, and extra large general-purpose VM instances and **high-CPU, medium** and **extra large** instances. It is possible that more instance types (**e.g., I/O high, memory high, storage high, etc.**) will be added in the future should a demand for them arise. Other IaaS providers—for example, GoGrid [3] and FlexiScale [4]—follow similar strategy.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- Infrastructure SLA's

- Thus, to deploy a service on a cloud, **a service provider orders suitable virtual hardware** and installs its application software on it.
- From the IaaS provider, a given service configuration is a virtual resource array of black box resources, which correspond to the number of instances of resource type.
- For example, a typical three-tier application may contain **ten general-purpose small instances** to run Web front-ends, **three large instances** to run an application server cluster with load balancing and redundancy, and **two large instances** to run a replicated database.
- A risk mitigation mechanism to protect user experience in the IaaS model is offered by infrastructure SLAs (i.e., the SLAs formalizing capacity availability) signed between service provider and IaaS provider.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- Infrastructure SLA's

- There is **no universal approach to infrastructure SLAs**. As the IaaS field matures and more experience is being gained, some methodologies may become more popular than others. Also some methods may be more suitable for specific workloads than others. There are three main approaches as follows.
- **No SLAs**. This approach is based on two premises: **(a)** Cloud always has spare capacity to provide on demand, and **(b)** services are not QoS sensitive and can withstand moderate performance degradation. This methodology is best suited for the best effort workloads.
- **Probabilistic SLAs**. These SLAs allow **us to trade capacity availability for cost of consumption**. Probabilistic SLAs specify clauses that determine availability percentile for contracted resources computed over the SLA evaluation period. **The lower the availability percentile, the cheaper the cost of resource consumption**. This type of SLA is **suitable for small and medium businesses and for many enterprise grade applications**.
- **Deterministic SLAs**. These are, in fact, probabilistic SLAs where **resource availability percentile is 100%**. These SLAs are most stringent and difficult to guarantee. From the provider's point of view, they do not admit capacity multiplexing. Therefore this is the most costly option for service providers, which may be applied for critical services.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- Infrastructure SLA's

- We will focus on probabilistic SLAs, however, because they represent the more interesting and flexible option and lay the foundation for the rest of discussion on **statistical multiplexing of capacity**.
- Before we can proceed, we need to define the concept, **elasticity rules**, which are scaling and de-scaling policies that guide transition of the service from one configuration to another to match changes in the environment. The main motivation for defining these policies stems from the pay-as-you-go billing model of IaaS clouds. The service owner is interested in paying only for what is really required to satisfy workload demands minimizing the over-provisioning overhead. There are **three types of elasticity rules**:
 - **Time-driven**: These rules **change the virtual resources array in response to a timer event**. These rules are useful for predictable workloads—for example, for services with well-known business cycles.
 - **OS Level Metrics-Driven**: These rules react on predicates defined in terms of the OS parameters (see Amazon Auto-scaling Service). These **auto-scaling policies are useful for transparently scaling and de-scaling services**. The problem is, however, that in many cases **this mechanism is not precise enough**.
 - **Application Metrics-Driven**: This is a unique RESERVOIR offering that **allows an application to supply application-specific policies** that will be transparently executed by IaaS middleware in reacting on the monitoring information supplied by the service-specific monitoring probes running inside VMs.

BITS Pilani

OpenNebula additional information available @

<https://www.youtube.com/watch?v=Q8wHwnsEkRI>

<http://opennebula.org/>

For additional information refer to **Chapter 6** (On the Management of Virtual Machines for Cloud Infrastructures) from the Textbook - Cloud Computing – Principles and Paradigms. John Wiley Pub, 2011

Rajkumar Buyya, James Broburg & Anderzej M.G,

Agenda



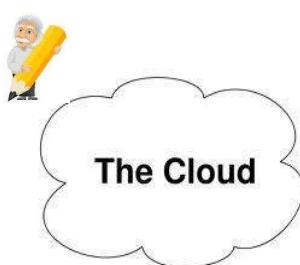
❖ Capacity Management Recap

- ❖ Multi Tenancy in the Cloud
- ❖ 4 Models of Multi Tenancy

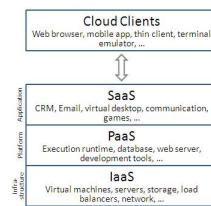
2

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

Anatomy of a Cloud Ecosystem



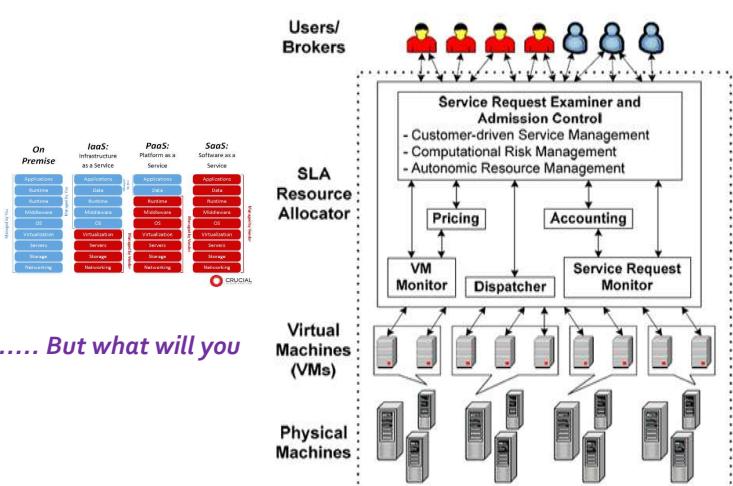
Is it So??????



CLOUD requires managing..... But what will you manage?

• Cloud Distributed environment

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



Importance of Capacity Management

When Capacity is Too Little (under capacity)

- Application Sizing not performed
- Controlled by limits
- Lower usage costs
- Service Levels affected – by how much?
- High impact on business?
 - Loss of service
 - SLA breach penalties



- Gartner – “most organizations overprovision their infrastructure by at least 100%”

When Capacity is Too Much (over capacity)

- Unlimited access to resources
- Service Levels unaffected
- Costs – higher resource usage, software licensing
- Impacts on other services
 - Poor performance
 - Wasted resources (multi virtual CPU (vSMP) & single-threaded applications)
 - VM Sprawl
 - Increased pressure to manage virtual resources



What are the Challenges?



Capacity on Demand-
Too Little, Too Much
or Just Right?



Automation and
Control



Security Protecting
your loved ones”



Scalability- In / Up or
Out?



Old Habits – The
Potential Risks



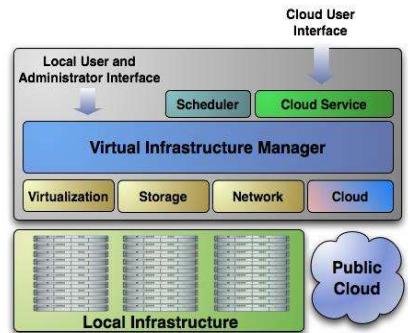
Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine’s software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.
- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization’s goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

What is a Virtual Infrastructure Manager?

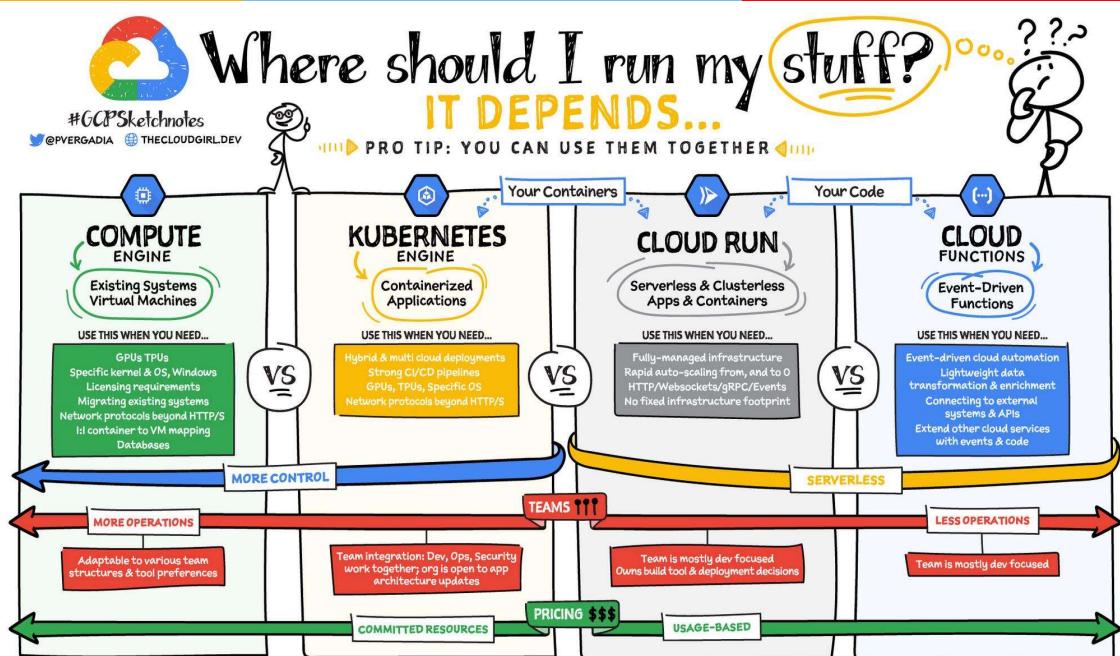


- A VIM runs on top of a hypervisor in a virtualized environment. The hypervisor allocates and manages virtual machines. The VIM deals with the allocation of resources in the virtual infrastructure. They include computational resources (processors), storage, and network resources. Virtual infrastructure management allows the allocation to happen based on current requirements, rather than being statically allocated.
- The VIM carries out several tasks:
 - Allocating resources in accordance with traffic engineering rules.
 - Support for defining operational rules.
 - Definition of hub-to-facility mapping.
 - Providing information for provisioning virtual infrastructure orchestration (VIO).



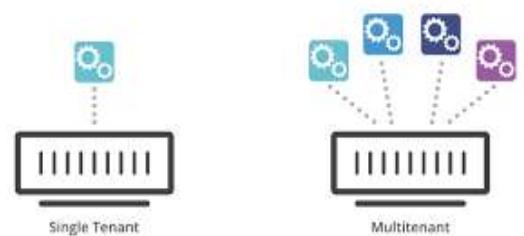
BITS Pilani

What is a Cloud Workload





Multi Tenancy



What is Multi Tenancy?

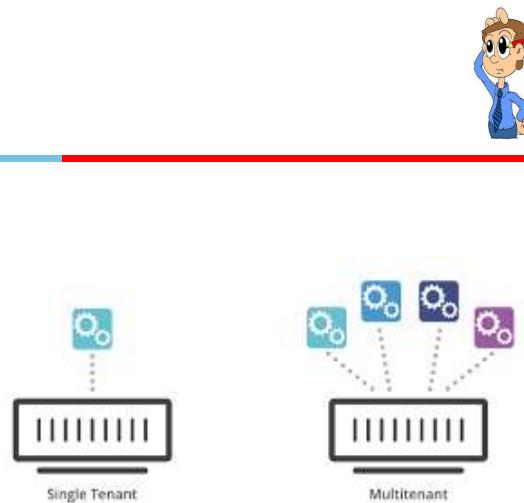


 Multi-tenancy is an architectural pattern

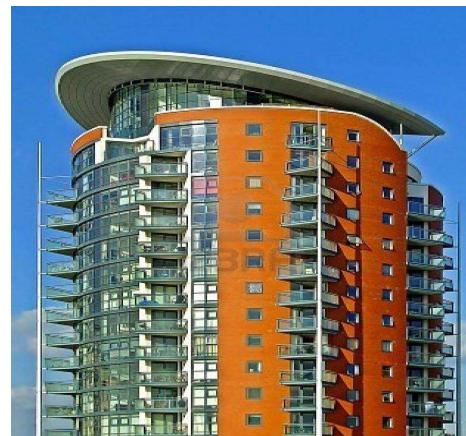
 A single instance of the software is run on the service provider's infrastructure

 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.



What is Multi Tenancy?

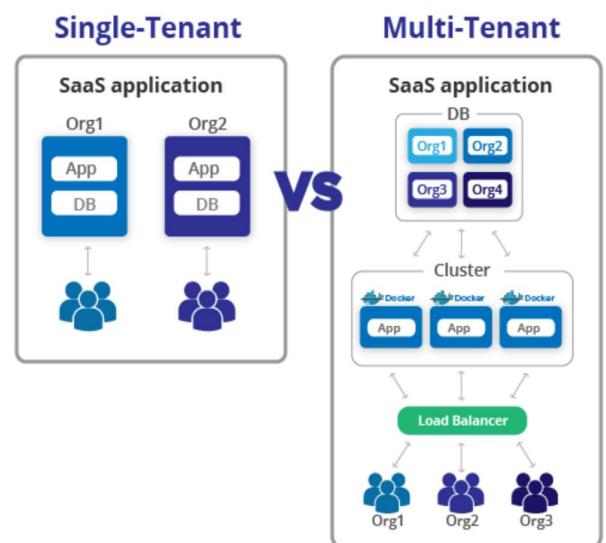


- Multi-tenancy is an architecture in which **a single instance of a software application** serves **multiple customers**. Each **customer** is called a **tenant**. Tenants may be given the ability **to customize** some parts of the application, such as color of the user interface (UI) or business rules, **but they cannot customize the application's code**.

BITS Pilani

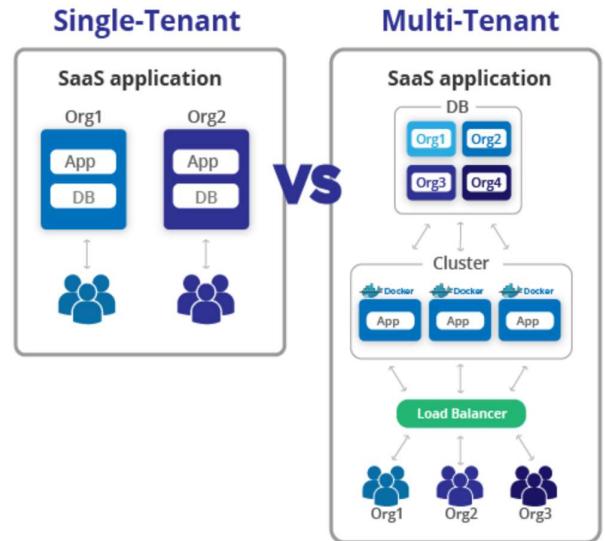
Some Facts

- In the **multi tenant architecture**, the **application** is **redesigned** to handle the resource sharing between the multiple tenants.
- For example, SalesForce.com (service provider) hosts the CRM application using their infrastructure.
- A company who wants to use this hosted CRM application for their business is the customer and the employees of the companies to whom the company provides privileges to access the CRM application are the actual users of the application



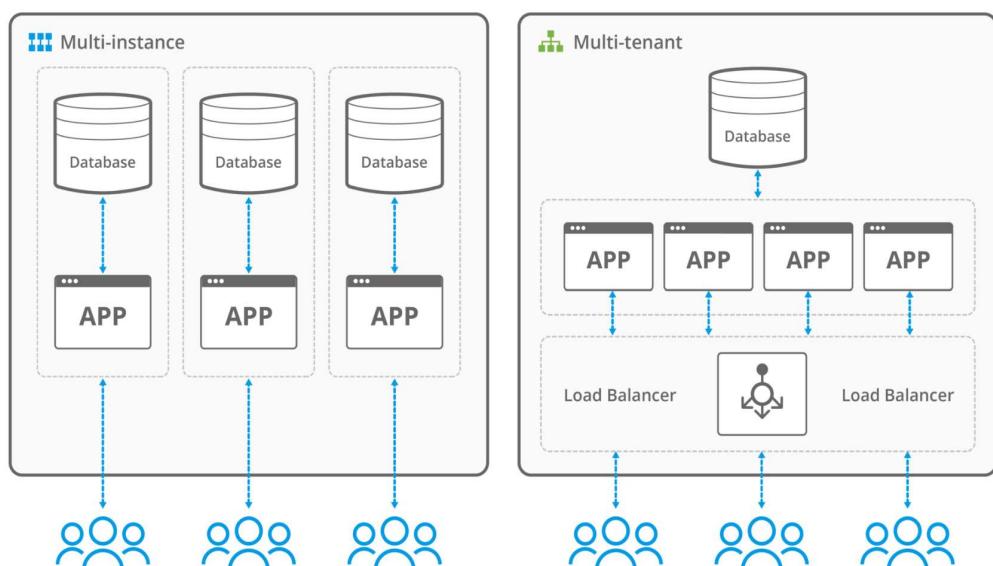
Some Facts

- With this architecture, data, configuration, user management, tenant specific functionality etc are shared between the tenants.
- MT contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.
- In virtualization, the user is given the illusion that he own the complete infrastructure on which application is running through concept of virtual machine.
- The hypervisor plays important role to achieve the separation between the multiple users.



BITS Pilani

Multi Tenant vs Multi Instance



IAAS – MT Model

In (IaaS), a single physical or virtual infrastructure is shared among multiple tenants. Each tenant is provided with its own logical environment, which includes compute resources such as virtual machines (VMs), storage, networking, and other infrastructure components. There are several ways to implement a multi-tenant deployment model for IaaS:

1. Virtual machine isolation: In this model, each tenant is provided with a set of virtual machines that are completely isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.

2. Network isolation: In this model, each tenant is provided with its own virtual network that is isolated from other tenants. This approach provides better performance and efficiency than virtual machine isolation, but may require more complex network management.

3. Resource pool isolation: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.

4. Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant.

For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

BITS Pilani

PAAS – MT Model

1. Application-level isolation: In this model, each tenant's application is isolated from other tenants' applications at the application level. This can be achieved through containerization or virtualization, and may also involve isolating the application's data and configuration settings.

2. Tenant-level isolation: In this model, each tenant is provided with a dedicated instance of the platform, which is isolated from other tenants. This approach provides strong security and isolation, but can be less efficient than other approaches.

3. Resource pool isolation: In this model, each tenant is provided with a dedicated set of compute resources such as CPU, memory, and storage that are isolated from other tenants. This approach provides good performance and resource allocation, but may be less secure than other approaches.

4. Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require strong security and isolation, while others may prioritize performance and efficiency.

SAAS – MT Model

There are several ways to implement a multi-tenant deployment model for SaaS:

1.Database-level isolation: In this model, each tenant's data is stored in a separate database schema, which provides complete isolation and security. This approach can be efficient and scalable, but may require complex database management.

2.Application-level isolation: In this model, each tenant's data is kept separate at the application level, which may involve partitioning data and configuration settings. This approach can be more flexible and easier to manage than the database-level isolation model, but may be less secure.

3.Hybrid model: In this model, a combination of the above models is used to meet the specific requirements of each tenant. For example, some tenants may require complete data isolation, while others may be willing to share some components of the application.

BITS Pilani

Benefits

Cost Savings –

- An application instance usually incurs a certain amount of memory and processing overhead which can be substantial when multiplied by many customers, especially if the customers are small.
- As the single instance is shared between multiple tenants this cost overhead can be segregated between multiple tenants.

Data aggregation–

- In non MT architecture, the data for different customers will be located in different database schemas and pulling information from all of them can be a very cumbersome task.
- In MT architecture, instead of collecting data from multiple data sources, with potentially different database schemas, all data for all customers is stored in a single database schema.
- Thus, running queries across customers, mining data, and looking for trends is much simpler.

Benefits

Release management –

- MT simplifies the release management process.
- In a traditional release management process, packages containing code and database changes are distributed to client desktop and/or server machines.
- These packages then have to be installed on each individual machine.
- With the multitenant model, the package typically only needs to be installed on a single server. This greatly simplifies the release management process.

Disadvantages

Complexity –

- Because of the additional customization complexity and the need to maintain per-tenant metadata, multitenant applications require a larger development effort.

Risks-

- At the same time, multi-tenancy increases the risks and impacts inherent in applying a new release version.
- As there is a single software instance serving multiple tenants, an update on this instance may cause downtime for all tenants even if the update is requested and useful for only one tenant.
- Also, some bugs and issues resulted from applying the new release could manifest in other tenants' personalized view of the application.
- Because of possible downtime, the moment of applying the release may be restricted depending on time usage schedule of more than one tenant.

Characteristics of MT Architecture

Customization –

- Multi-tenant applications are typically required to provide a high degree of customization to support each target organization's needs. Customization typically includes the following aspects:
 - ❖ **Branding:** allowing each organization to customize the look-and-feel of the application to match their corporate branding (often referred to as a distinct "skin").
 - ❖ **Workflow:** accommodating differences in workflow to be used by a wide range of potential customers.
 - ❖ **Extensions to the data model:** supporting an extensible data model to give customers the ability to customize the data elements managed by the application to meet their specific needs.
 - ❖ **Access control:** letting each client organization independently customize access rights and restrictions for each user.

Quality of service

- Multitenant applications are expected to provide adequate isolation of security, robustness and performance between multiple tenants which is provided by the layers below the application in case of multi-instance applications

BITS Pilani

MT- Different Levels

- Implementing the highest degree of resource sharing for all resources may be prohibitively expensive in development effort and complexity of the system.
- A balanced approach, where there is fine grained sharing of resources only for important resources, may be the optimum approach.
- The four levels of multi-tenancy are described in the following list for any given resource in a cloud system, the appropriate level could be selected
 - **Custom instances**
 - **Configurable instances**
 - **Configurable, multi-tenant efficient instances**
 - **Scalable, configurable, multi tenant efficient resources**

MT- Different Levels

Custom instances

- Lowest level of MT
- Each customer has own custom version of application
- Different versions of application are running differently
- Extremely difficult to manage as needs dedicated support for each customer

Configurable instances

- Same version of application is shared between the customers with customizations specific to each customer
- Different instances of same application are running
- Supports customization like logos on the screen, tailor made workflows
- Managing application is better than custom instances approach as only one copy needs to be managed
- Upgrades are simple and seamless

BITS Pilani

MT- Different Levels

Configurable, multi-tenant efficient instances

- Same version of application is shared between the customers through a single instance of application
- More efficient usage of the resources
- Management is extremely simple

Scalable, configurable, multi tenant efficient resources

- All features of level 3 are supported.
- Application instances are installed on cluster of computers allowing it to scale as per demand.
- Maximum level of resource sharing is achieved.
- Example, Gmail

BITS Pilani

Security in MT

- The key challenge in multi-tenancy is the secure sharing of resources. A very important technology to ensure this is authentication.
- Clearly each tenant would like to specify the users who can log in to the cloud system. Unlike traditional computer systems, the tenant would specify the valid users, but authentication still has to be done by the cloud service provider.
- Two basic approaches can be used: **a centralized authentication system** or a **decentralized authentication system**.
- In the **centralized system**, all authentication is performed using a centralized user data base. The cloud administrator gives the tenant's administrator rights to manage user accounts for that tenant. When the user signs in, they are authenticated against the centralized database.

BITS Pilani

Security in MT

- In the **decentralized system**, each tenant maintains their own user data base, and the tenant needs to deploy a federation service that interfaces between the tenant's authentication framework and the cloud system's authentication service.
- **Decentralized authentication** is useful if **single sign-on is important**, since centralized authentication systems will require the user to sign on to the central authentication system in addition to signing on to the tenant's authentication system.
- However, **decentralized authentication systems** have the **disadvantage that they need a trust relationship between the tenant's authentication system and the cloud provider's authentication system**. Given the self-service nature of the cloud (i.e., it is unlikely that the cloud provider would have the resources to investigate each tenant, and ensure that their authentication infrastructure is secure), centralized authentication seems to be more generally applicable.

Multi Tenancy: Resource Sharing

- Two major resources that need to be shared are storage and servers.
- The basic principles for sharing of these resources are described first.
- This is followed by a deeper discussion that focuses on the question of how these resources can be shared at a fine granularity, while allowing the tenants to customize the data to their requirements.

Sharing storage resources:

- In a multi-tenant system, many tenants share the same storage system. Cloud applications may use two kinds of storage systems: File systems and databases, where the term database is used to mean not only relational databases, but NoSQL databases as well.
- The discussion is focused on sharing data for different users in a database. The focus is also on multi-tenant efficient approaches where there is only one instance of the database which is shared among all the tenants

Resource Sharing Approaches

STORAGE Sharing Scenarios in MT

Dedicated tables per tenant

- We discuss only the approach where only one instance of database is shared between the tenants.
- Each tenant has separate copy of table
- Only tenant is given the privileges to access these tables, no other can access it
- Customizations can be easily added to the tenant's tables

Shared table approach

- Tables are shared between the tenants
- Needs to isolate between the tenants data in different rows using the unique tenant id assigned to each tenant
- More space efficient than dedicated table approach
- Needs more compute resources as it needs to use view to make join to retrieve tenant specific data
- Metadata table needs to be maintained for tenant's information
- Managing customization is difficult

Tenant 1 Employee Table			
Empld	EmpName	EmpDept	Salary
1	P	IT	10
2	Q	Sales	20
3	R	IT	30

Tenant 2 Employee Table			
Empld	EmpName	EmpDept	Salary
11	A	Manu	234
22	B	Sales	245
33	c	Retail	335

Data Table				
Tenant Id	Empld	EmpName	EmpDept	Salary
1	1	P	IT	10
1	2	Q	Sales	20
1	3	R	IT	30
2	11	A	Manu	234
2	22	B	Sales	245
2	33	C	Retail	335

Metadata table	
Tenant Id	Tenant Name
1	Tenant1
2	Tenant2

Support for Customization

Customization:

- It is important for the cloud infrastructure to support customization of the stored data, since it is likely that different tenants may want to store different data in their tables.
- For example, an automobile repair shop, different shops may want to store different details about the repairs carried out.
- Three methods for doing this are described in the next slides. It is to be noted that difficulties for customization occur only in the shared table method.
- In the dedicated table method, each tenant has their own table, and therefore can have different schema.

Support for Customization

Pre Allocated Columns:

- In shared table approach, it's very complex to provide support for the customizations.
- Each tenant might have his unique requirements to store data in the tables and using shared table approach, managing such requirements needs to come up with proper data architecture.

Pre allocated columns

- Fixed number of columns is reserved for custom columns
- If the numbers of custom column are too less than the reserved custom columns then space are wasted
- If the numbers of custom columns are more than the reserved custom columns then customer will feel restricted.

Data table 1

Tenant Id	EmpId	EmpName	EmpDept	Salary	Custom1	Custom2
1						
1						
2						

Metadata table

Tenant Id	Tenant Name	Custom1 Name	Custom1 Type	Custom2 Name	Custom2 Type
1	Tenant1	Country	String	CurrencyUnit	String
2	Tenant2	Joining Date	Date	Region	String

Support for Customization

Name-Value Pairs:

- The major problem with the pre-allocated columns technique is that there could be a lot of wasted space.
- If the number of columns is too low, then users will feel constrained in their customizations.
- However, if the number is too big, there will be a lot of space wastage.

Name-Value Pairs

- A metadata table for tenant is maintained
- A data table is specified with standard common columns and has extra column at end to point to another name value pair table
- Name value pair table (aka pivot table) actually includes the custom fields for this record.
- The actual custom fields are stored with data type and other metadata information.
- Space efficient as compared to pre allocated columns method
- Joins are involved to fetch tenant specific data

Data table 1

Tenant Id	Car license	Service	Cost	Name-value pair rec
1				275
2				
2				
1				
3				
2				

Data table 2 (name-value pairs)

Name-value pair rec	NameID	Value
275	15	5.5

Metadata table 1

Name Id	Name	Type
15	Service rating	int
	Service manager	string

Metadata table 2

Tenant Id	Data
1	Best garage
2	Friendly garage
3	Honest garage

XML method :

- The last column of database table is reserved for storing the information in XML format

Support for Customization

- Let's consider Tenants have following table structure that needs to be mapped to the Name-Value pair table structure.
- Each tenant table has:
 - standard common fields and
 - few custom fields having varying data type

Tenant1 Employee table

Empld	EmpName	Salary	Country (Custom)	Currency(Custom)
1	Pravin	10	India	Rs
2	Prashnat	20	China	Yen

Tenant2 Employee table

Empld	EmpName	Salary	JoiningDate(Custom)	Region (String)
11	Ravi	30	1 Jan 2012	MH
22	Aakash	40	1 Apr 2012	KA

Sample MT Architecture

MT data architecture –

Metadata table 1 (Tenant Table)

Tenant Id	Tenant Name	Description
1	Tenant1	Tenant1 Employee table
2	Tenant2	Tenant2 Employee table

Metadata table 2 (Fields / Columns table)

Field ID	Field Name	Field Datatype
Field1	Country	String
Field2	Currency	String
Field3	JoiningDate	Date
Field4	Region	String

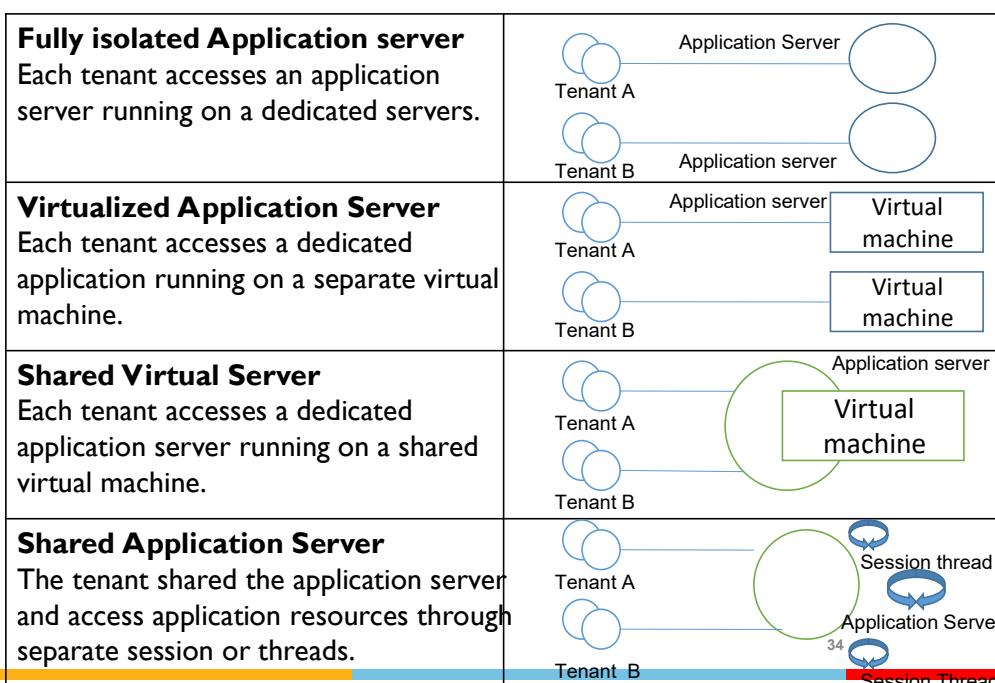
Data table 1

Tenant Id	Empld	EmpName	Salary	Name-Value Pair Record id
1	1	Pravin	10	101
1	2	Prashant	20	102
2	11	Ravi	30	103
2	22	Aakash	40	104

Data table2 (Name – Value Pairs table)

Name-Value Pair Record id	Field ID	Value
101	Field1	India
101	Field2	Rs
102	Field1	China
102	Field2	Yen
103	Field3	1 Jan 2012
103	Field4	MH
104	Field3	1 Apr 2012
104	Field4	KA

Multi Tenancy: Resource Sharing



Did You Know ?

Here are some interesting facts about multi-tenancy in the cloud:

1. Multi-tenancy is one of the key features of cloud computing that makes it so attractive to businesses. It allows multiple customers to share the same infrastructure and services, which can lead to cost savings and improved efficiency.
2. One of the biggest challenges of multi-tenancy is ensuring that each tenant's data is kept separate and secure. This requires careful planning and implementation of security measures to prevent data leakage and unauthorized access.
3. Multi-tenancy can be implemented at different levels in the cloud, including infrastructure, platform, and software. Each level requires different techniques and technologies to ensure proper isolation and security.
4. In the public cloud model, multi-tenancy is typically implemented using virtualization and resource sharing. Each tenant is assigned their own virtual resources, which are isolated from other tenants.

BITS Pilani

Did You Know ?

1. In the private cloud model, multi-tenancy can be implemented using virtualization or containerization, and is typically achieved through strict access controls and resource partitioning.
2. Hybrid cloud environments can offer the benefits of both public and private cloud models, but can also add complexity and require careful management to ensure proper isolation and security.
3. Multi-tenancy can offer significant cost savings for businesses, as they only pay for the resources they use. This can also lead to better resource utilization and scalability, as resources are shared among multiple tenants.
4. Multi-tenancy is not suitable for all types of applications, and some applications may require dedicated resources and environments. It is important to carefully evaluate the requirements of each application and choose the appropriate deployment model.

Overall, multi-tenancy is a key feature of cloud computing that can offer significant benefits to businesses, but it requires careful planning and implementation to ensure proper isolation and security.

SEAY-ZG527 CLOUD COMPUTING

Session 11: 05-05-2018
This session Cloud Security Issues & Solution



Today's Topics

-
- Cloud Security Issues



SECURITY

Cloud Security Issues : Introduction



Introduction

- Cloud Ecosystem : A massive scale of Virtual Resources
- Also a massive concentration of risk
 - expected loss from a single breach can be significantly larger
 - concentration of “users” represents a concentration of threats
- “Ultimately, you can outsource responsibility but you can’t outsource accountability.”

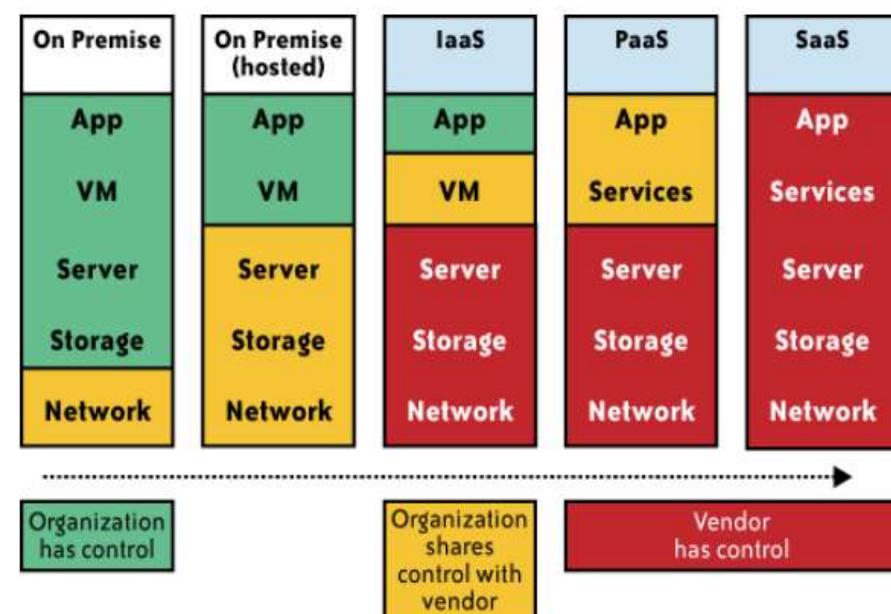


Cloud Computing: Security Analysis?

- Cloud computing definitely makes sense if your own security is weak, missing features, or below average.
- Ultimately, if
 - the cloud provider's security people are “better” than yours (and leveraged at least as efficiently),
 - the web-services interfaces don't introduce too many new vulnerabilities, and
 - the cloud provider aims at least as high as you do, at security goals,
- then cloud computing has better security.

From [2] John McDermott, ACSAC 09

Impact of cloud on the governance structure of IT organizations



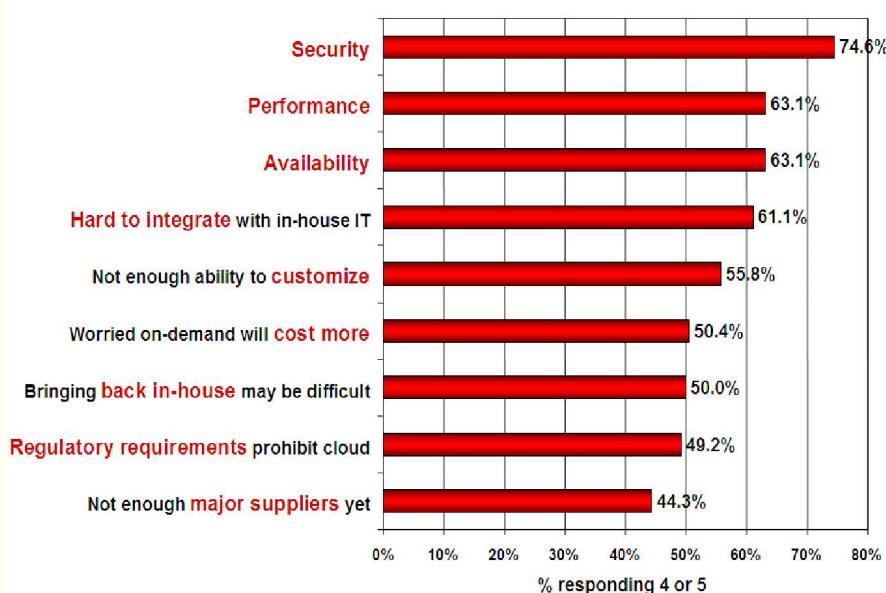
Cloud Adoption : Reluctance

- 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

7

Companies are afraid to use clouds

Q: Rate the challenges/issues ascribed to the 'cloud'/on-demand model
(1=not significant; 5=very significant)



[Chow09ccsw]

8

Source: IDC Enterprise Panel, August 2008 n=244

Cloud Security Issues

Most security problems stem from:

- Loss of control
- Lack of trust (mechanisms)
- Multi-tenancy

These problems exist mainly in rdparty management models

Self-managed clouds still have security issues, but not related to above

[Chow09ccsw]

9

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

Lack of Trust in the Cloud

- Trusting a third party requires taking risks
- Defining trust and risk
 - Opposite sides of the same coin (J. Camp)
 - People only trust when it pays (Economist's view)
 - Need for trust arises only in risky situations
- Defunct third party management schemes
 - Hard to balance trust and risk
 - e.g. Key Escrow (Clipper chip)
 - Is the cloud headed toward the same path?

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 - CIA

- 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?
- 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

13 From [5] www.cs.jhu.edu/~ragib/sp10/cs412

Taxonomy of Fear (cont.)

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

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

14

Taxonomy of Fear (cont.)

- Auditability and forensics (out of control of data)
 - Difficult to audit data held outside organization in a cloud
 - Forensics also made difficult since now clients don't maintain data locally
- Legal dilemma 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?

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

15

Threat Model

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

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

16

Threat Model

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

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

17

What is the issue?

- The core issue here is the levels of trust
 - Many cloud computing providers trust their customers
 - Each customer is physically commingling its data with data from anybody else using the cloud while logically and virtually you have your own space
 - The way that the cloud provider implements security is typically focused on the fact that those outside of their cloud are evil, and those inside are good.
- But what if those inside are also evil?

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

18

Attacker Capability: Malicious Insiders

- At client
 - Learn passwords/authentication information
 - Gain control of the VMs
- At cloud provider
 - Log client communication
 - Can read unencrypted data
 - Can possibly peek into VMs, or make copies of VMs
 - Can monitor network communication, application patterns
 - Why?
 - Gain information about client data
 - Gain information on client behavior
 - Sell the information or use itself

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

19

Attacker Capability: Outside attacker

- What?
 - Listen to network traffic (passive)
 - Insert malicious traffic (active)
 - Probe cloud structure (active)
 - Launch DoS
- Goal?
 - Intrusion
 - Network analysis
 - Man in the middle
 - Cartography

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

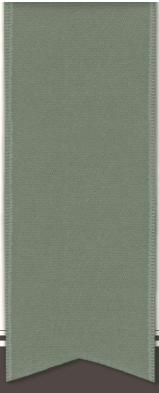
20

Challenges for the attacker

- How to find out where the target is located?
- How to be co-located with the target in the same (physical) machine?
- How to gather information about the target?

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

21



PART II: SECURITY AND PRIVACY ISSUES IN CLOUD COMPUTING - BIG PICTURE

From [6] Cloud Security and Privacy by Mather and Kumaraswamy

22

Data Security and Storage

- Several aspects of data security, including:
 - Data-in-transit
 - Confidentiality + integrity using secured protocol
 - Confidentiality with non-secured protocol and encryption
 - Data-at-rest
 - Generally, not encrypted , since data is commingled with other users' data
 - Encryption if it is not associated with applications?
 - But how about indexing and searching?
 - Processing of data, including multitenancy
 - For any application to process data

From [6] Cloud Security and Privacy by Mather and Kumaraswamy

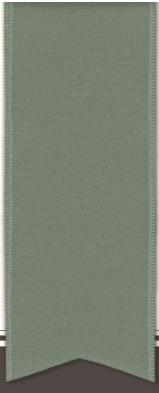
23

What is Privacy?

- The concept of privacy varies widely among (and sometimes within) countries, cultures, and jurisdictions.
- It is shaped by public expectations and legal interpretations;
 - as such, a concise definition is elusive if not impossible.
- Privacy rights or obligations are related to the collection, use, disclosure, storage, and destruction of personal data
- At the end of the day, privacy is about the accountability of organizations to data subjects, as well as the transparency to an organization's practice around personal information.

From [6] Cloud Security and Privacy by Mather and Kumaraswamy

24



PART III. POSSIBLE SOLUTIONS

25

Security Issues in the Cloud

- In theory, minimizing any of the issues would help Third Party Cloud Computing
 - 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)
 - 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

Minimize Lack of Trust: Policy Language

- Consumers have specific security needs but don't have a say-so in how they are handled
 - Currently consumers cannot dictate their requirements to the provider (SLAs are one-sided)
- Standard language to convey one's policies and expectations
 - Agreed upon and upheld by both parties
 - Standard language for representing SLAs
- Create policy language with the following characteristics:
 - Machine-understandable (or at least processable),
 - Easy to combine/merge and compare

Minimize Lack of Trust: Certification

- Certification
 - Some form of reputable, independent, comparable assessment and description of security features and assurance
 - Sarbanes-Oxley, DIACAP, DISTCAP, etc
- Risk assessment
 - Performed by certified third parties
 - Provides consumers with additional assurance

Minimize Loss of Control: Monitoring

- Cloud consumer needs situational awareness for critical applications
 - When underlying components fail, what is the effect of the failure to the mission logic
 - What recovery measures can be taken
 - by provider and consumer
- Requires an application-specific run-time monitoring and management tool for the consumer
 - The cloud consumer and cloud provider have different views of the system
 - Enable both the provider and tenants to monitor the components in the cloud that are under their control

Minimize Loss of Control: Monitoring (Cont.)

- Provide mechanisms that enable the provider to act on attacks he can handle.
 - infrastructure remapping
 - create new or move existing fault domains
 - shutting down offending components or targets
 - and assisting tenants with porting if necessary
 - Repairs
- Provide mechanisms that enable the consumer to act on attacks that he can handle
 - application-level monitoring
 - RAdAC (Risk-adaptable Access Control)
 - VM porting with remote attestation of target physical host
 - Provide ability to move the user's application to another cloud

Minimize Loss of Control: Utilize Different Clouds

- The concept of 'Don't put all your eggs in one basket'
 - Consumer may use services from different clouds through an intra-cloud or multi-cloud architecture
 - A multi-cloud or intra-cloud architecture in which consumers
 - Spread the risk
 - Increase redundancy (per-task or per-application)
 - Increase chance of mission completion for critical applications
- Possible issues to consider:
 - Policy incompatibility (combined, what is the overarching policy?)
 - Data dependency between clouds
 - Differing data semantics across clouds
 - Knowing when to utilize the redundancy feature
 - monitoring technology
 - Is it worth it to spread your sensitive data across multiple clouds?
 - Redundancy could increase risk of exposure

Minimize Loss of Control: Access Control

- Many possible layers of access control
 - E.g. access to the cloud, access to servers, access to services, access to databases (direct and queries via web services), access to VMs, and access to objects within a VM
 - Depending on the deployment model used, some of these will be controlled by the provider and others by the consumer
- Regardless of deployment model, provider needs to manage the user authentication and access control procedures (to the cloud)
 - Federated Identity Management: access control management burden still lies with the provider
 - Requires user to place a large amount of trust on the provider in terms of security, management, and maintenance of access control policies.
 - This can be burdensome when numerous users from different organizations with different access control policies, are involved

Minimize Multi-tenancy

- Can't really force the provider to accept less tenants
 - Can try to increase isolation between tenants
 - Strong isolation techniques (VPC to some degree)
 - QoS requirements need to be met
 - Policy specification
 - Can try to increase trust in the tenants
 - Who's the insider, where's the security boundary? Who can I trust?
 - Use SLAs to enforce trusted behavior

Conclusion

- Cloud computing is sometimes viewed as a reincarnation of the classic mainframe client-server model
 - However, resources are ubiquitous, scalable, highly virtualized
 - Contains all the traditional threats, as well as new ones
- In developing solutions to cloud computing security issues it may be helpful to identify the problems and approaches in terms of
 - Loss of control
 - Lack of trust
 - Multi-tenancy problems

Agenda



❖ Multi Tenency Recap

- ❖ SLA Management
- ❖ Introduction to SLA
- ❖ Types of SLA
- ❖ SLO & SLA
- ❖ SLA Life Cycle

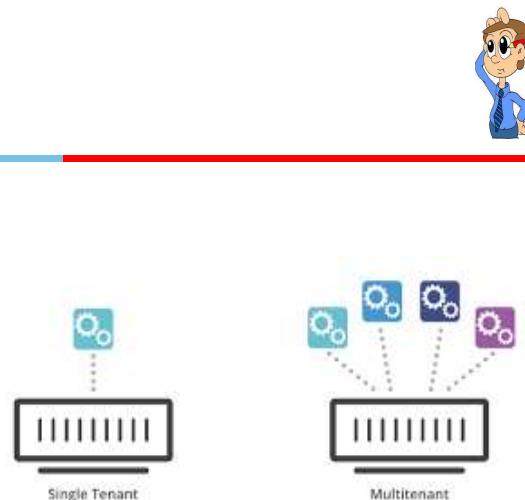
2

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

What is Multi Tenancy?

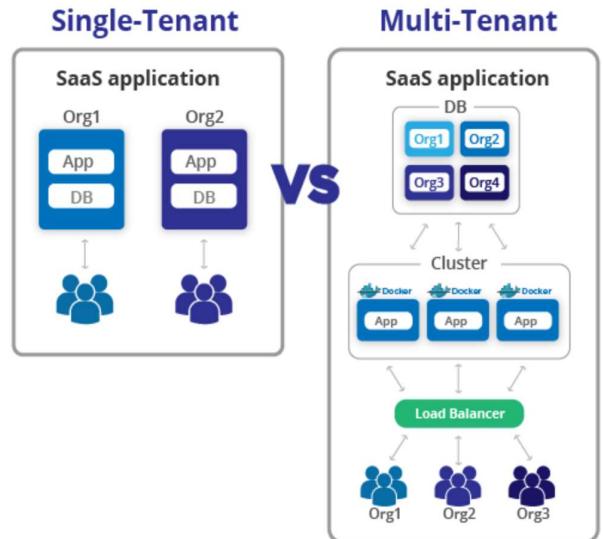


	Multi-tenancy is an architectural pattern
	A single instance of the software is run on the service provider's infrastructure
	Multiple tenants access the same instance.
	In contrast to the multi-user model, multi-tenancy requires customizing the single instance according to the multi-faceted requirements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



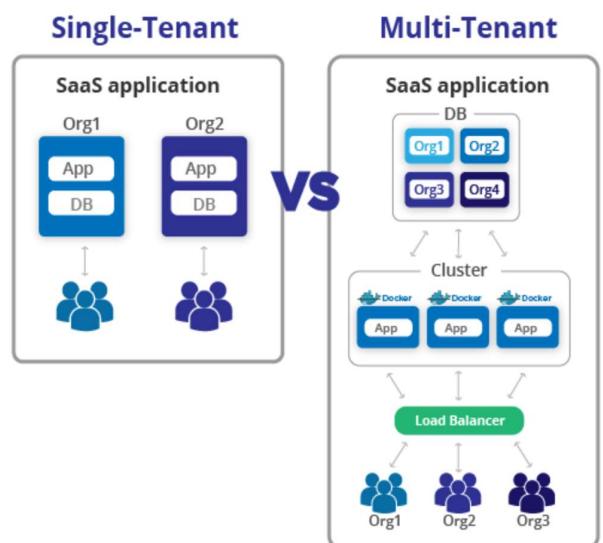
Some Facts

- In the multi tenant architecture, the application is redesigned to handle the resource sharing between the multiple tenants.
- For example, SalesForce.com (service provider) hosts the CRM application using their infrastructure.
- A company who wants to use this hosted CRM application for their business is the customer and the employees of the companies to whom the company provides privileges to access the CRM application are the actual users of the application

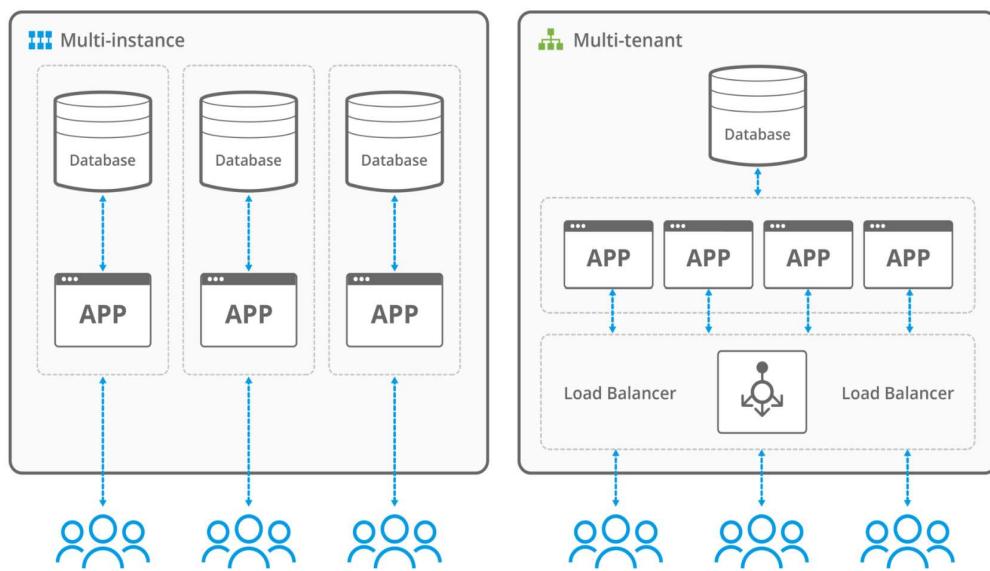


Some Facts

- With this architecture, data, configuration, user management, tenant specific functionality etc are shared between the tenants.
- MT contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.
- In virtualization, the user is given the illusion that he own the complete infrastructure on which application is running through concept of virtual machine.
- The hypervisor plays important role to achieve the separation between the multiple users.



Multi Tenant vs Multi Instance



BITS Pilani

MT- Different Levels

Custom instances

- Lowest level of MT
- Each customer has own custom version of application
- Different versions of application are running differently
- Extremely difficult to manage as needs dedicated support for each customer

Configurable instances

- Same version of application is shared between the customers with customizations specific to each customer
- Different instances of same application are running
- Supports customization like logos on the screen, tailor made workflows
- Managing application is better than custom instances approach as only one copy needs to be managed
- Upgrades are simple and seamless

MT- Different Levels

Configurable, multi-tenant efficient instances

- Same version of application is shared between the customers through a single instance of application
- More efficient usage of the resources
- Management is extremely simple

Scalable, configurable, multi tenant efficient resources

- All features of level 3 are supported.
- Application instances are installed on cluster of computers allowing it to scale as per demand.
- Maximum level of resource sharing is achieved.
- Example, Gmail

BITS Pilani



BITS Pilani

Pilani | Dubai | Goa | Hyderabad



SLA Management



What is SLA or Service Level Agreement

- Describes a set of non functional requirements of the service.
- Example : RTO time – Return to Operation Time if case of failure



- SLO – Service Level Objective. That is, the objective to be achieved.
- KPI – Key Performance Indicators
- **Service Level Objective:**
- Objective of service quality that has to be achieved.
- Set of measurable KPIs with thresholds to decide if the objective is fulfilled or not.
 - Attainable
 - Repeatable
 - Measurable
 - Understandable
 - Meaningful
 - Controllable
 - Affordable
 - Mutually acceptable

BITS Pilani

SLA Role in High Availability

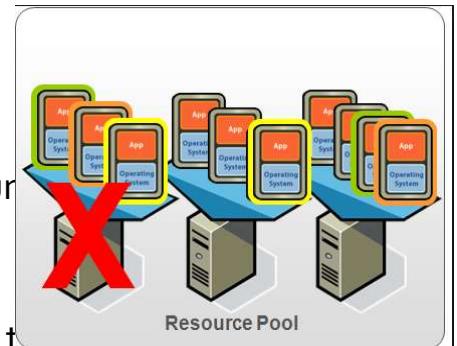
- **What is High Availability**
- Driven by SLA (Service Level Agreement)
- High Availability must conform to SLA. Goal here is to meet promised quality
- Examples: Service is available 99.95%.
 - Net Banking : Guarantees banking 24x7 . There are published down times called "SYSTEM MAINTENANCE" window
 - Duronto Express : Promises point to point service with only Service halts



SLA Role in High Availability

- **What deters HA**

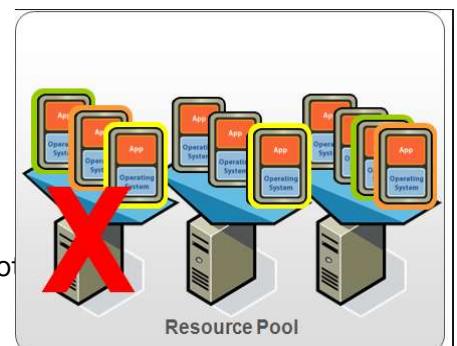
- Service outage which was unplanned because of Server or Human Error
- Service Disruption by Planned Maintenance windows (Downtime)
- Service Performance degradation due to lack of infrastructure or failure of critical components during peak load time
- Bottom Line : Need effective Capacity Planning to meet promised QoS or in other words maintain HA thus adhering to the SLA's



BITS Pilani

Steps for HA

- **Steps to achieve high availability**
- **Build for server failure**
 - Have redundant servers which can be made online
- **Build for zone failure**
 - Having DR sites in case of failure to switch over to backup site
- **Build for Cloud failure**
 - Plan for your Cloud setup to be robust and contained. Errors should not cascade, having fire door policy to contain threats or errors which affect the ecosystem
- **Automating and testing**
 - Test & Test again, low manual interference



The 3 Initialisms



BITS Pilani

SLA vs SLO

- The SLA is the entire agreement that specifies
- what service is to be provided,
- how it is supported,
- times,
- locations,
- costs,
- performance, and
- responsibilities of the parties involved.

SLOs are specific measurable characteristics of the SLA such as availability, throughput, frequency, response time, or quality.

SLIs are actual measure of the SLO and serves as a benchmark to compare against the promised SLO availability, throughput, frequency, response time, or quality.

The 3 Initialisms



Google Cloud

SRE implements DevOps

Presents:

SLIs, SLOs, SLAs, oh my!



Seth Vargo
@sethvargo



Liz Fong-Jones
@lizthegrey

BITS Pilani

SLA

- In the early days of web-application deployment, **performance of the application** at peak load was a single important criterion for provisioning server resources.
- **Provisioning** in those days involved deciding hardware configuration, determining the number of physical machines, and acquiring them upfront so that the overall business objectives could be achieved.
- The web applications were **hosted** on these dedicated individual servers within enterprises' own server rooms. These web applications were used to provide different kinds of e-services to various clients.

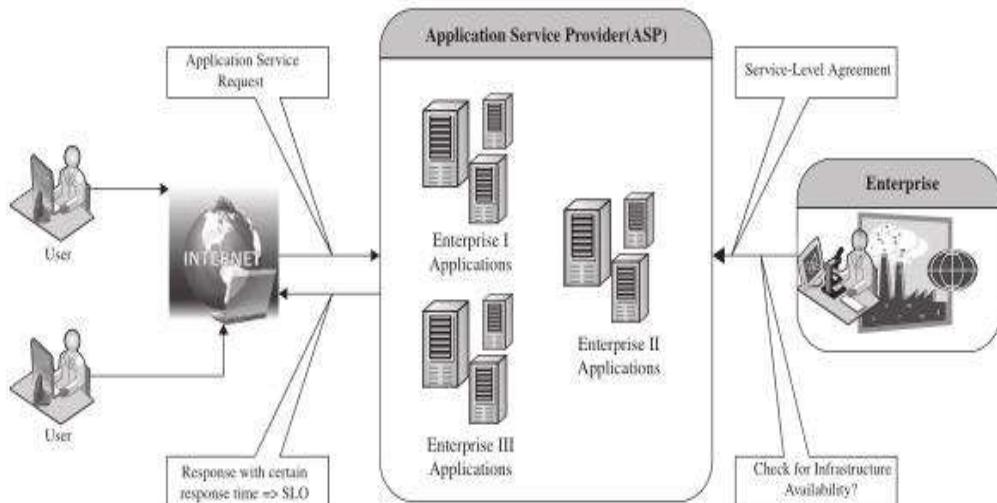


SLA

- Due to the increasing **complexity of managing the huge Data centres**, enterprises started outsourcing the application hosting to the infrastructure providers. They would procure the hardware and make it available for application hosting.
- It necessitated the enterprises to enter into a **legal agreement with the infrastructure service providers** to guarantee a minimum quality of service (QoS).
- Typically, the **QoS parameters** are related to the availability of the system CPU, data storage, and network for efficient execution of the application at peak loads.
- This legal agreement is known as the service-level agreement (SLA)



SLA



Co-Hosting Application

The dedicated hosting practice resulted in massive redundancies within the ASP's data centers due to the **underutilization of many of their servers**.

This is because the applications were not fully utilizing their servers' capacity at nonpeak loads.

To reduce the redundancies and increase the server utilization in data centers, ASPs started co-hosting applications with complementary work load patterns.

Co-hosting of applications means deploying more than one application on a single server. This led to further cost advantage for both the ASPs and enterprises.



BITS Pilani

Co-Hosting Application: Issues

However, newer challenges such as **application performance isolation** and **security guarantees have emerged** and needed to be addressed.

Performance isolation implies that one application should not steal the resources being utilized by other co-located applications.

For example, assume that application A is required to use more quantity of a resource than originally allocated to it for duration of time t. For that duration the amount of the same resource available to application B is decreased. This could adversely affect the performance of application B.

Security guaranty: Similarly, one application should not access and destroy the data and other information of co-located applications. Hence, appropriate measures are needed to guarantee security and performance isolation.

Co-Hosting Application: Solution

These challenges prevented ASPs from fully realizing the benefits of co-hosting. Virtualization technologies have been proposed to overcome the above challenges. The ASPs could exploit the containerization features of virtualization technologies to provide performance isolation and guarantee data security to different co-hosted applications.

The applications, instead of being hosted on the physical machines, can be encapsulated using virtual machines.

System resource allocation to these virtual machines can be made in two modes:

- (1) Conserving
- (2) Nonconserving.

In the **conserving mode**, a virtual machine demanding more system resources (CPU and memory) than the specified quota cannot be allocated the spare resources that remain un-utilized by the other co-hosted virtual machines.

In the **non-conserving mode** the spare resources that are not utilized by the co-hosted virtual machines can be used by the virtual machine needing the extra amount of resource. If the resource requirements of a virtual machine cannot be fulfilled from the current physical host, then the virtual machine can be migrated to another physical machine capable of fulfilling the additional resource requirements.

BITS Pilani

Traditional SLO Management Approaches

Load balancing – is to distribute the incoming request onto a set of physical machines, each hosting a replica of an application, so that the load on the machines is equally distributed. Front end node (node facing the client) receives the incoming requests and distributes these requests to different physical machines for further execution. Back end nodes serve the incoming requests.

Class agnostic load balancing – the front end machine are agnostic to nature of request. This means that the front end machine is neither aware of the type of client from which the request originates nor aware of the request category.

Class aware load balancing – The front end node additionally inspect the type of client making the request and/or the type of service requested before deciding which back end node should service the request.

Traditional SLO Management Approaches

Admission Control – These algorithms play an important role in deciding the set of requests that should be admitted into the application server when the server experiences very heavy loads. The objective of admission control mechanisms is to police the incoming requests and identify when the system faces overload situations.

Request based algorithms – reject new requests if the servers are running to their capacity. The disadvantage is that client's session may consist of multiple requests that are not necessarily unrelated. Some requests are rejected even if there are others that are honored.

Session based algorithms – ensure that longer sessions are completed and any new sessions are rejected. Once a session is admitted into the server, all future requests belonging to that session are admitted as well, even though new sessions are rejected by the system.

SLA Types

Infrastructure SLA – Infrastructure provider manages and offers guarantees on availability of the infrastructure, namely server machine, power, network connectivity and so on. Enterprises manage their applications that are deployed on the server machines. The machines are leased to customers and are isolated from machines of other customers.

For example, SLAs can be

Hardware availability – 99% uptime in a calendar month

Power availability – 99.99% of the time in calendar month

Data center network availability – 99.99% of the time in calendar month

Application SLA – In application co-hosting model, the server capacity is available to the applications based solely on their resource demands. Hence the service providers are flexible in allocating and de-allocating computing resources among the co-located applications. Therefore the service providers are also responsible for ensuring to meet their customers' application SLOs.

For example, SLAs can be

Web site response time (max of 3.5 sec per user request), Latency of web server (max of 0.2 sec per request), Latency of DB (max of 0.5 sec per query)

Infrastructure SLA vs Capacity Management

If temporal behavior of services with respect to resource demands is highly predictable, then capacity can be efficiently scheduled using reservations.

For **less predictable elastic workloads**, exact scheduling of capacity may not be possible.

Rather than that, **capacity planning and optimizations are required**.

- IaaS providers perform two complementary management tasks:

- (1) **Capacity planning** to make sure that SLA obligations are met as contracted with the service providers and;
- (2) **Continuous optimization** of resource utilization in specific workload to make the most efficient use of the existing capacity.

Infrastructure SLA

Thus, to deploy a service on a cloud, a **service provider orders suitable virtual hardware** and installs its application software on it.

From the IaaS provider, a given service configuration is a virtual resource array of black box resources, which correspond to the number of instances of resource type.

For example, a typical three-tier application may contain **ten general-purpose small instances** to run Web front-ends, **three large instances** to run an application server cluster with load balancing and redundancy, and **two large instances** to run a replicated database.

A **risk mitigation mechanism to protect user experience in the IaaS model is offered by infrastructure SLAs (i.e., the SLAs formalizing capacity availability)** signed between service provider and IaaS provider.

Infrastructure SLA

There is **no universal approach to infrastructure SLAs**. As the IaaS field matures and more experience is being gained, some methodologies may become more popular than others. Also some methods may be more suitable for specific workloads than others. There are three main approaches as follows.

No SLAs. This approach is based on two premises:

(a) Cloud always has spare capacity to provide on demand, and

(b) services are not QoS sensitive and can withstand moderate performance degradation. This methodology is best suited for the best effort workloads.

Probabilistic SLAs. (Epistemic) These SLAs allow us to trade capacity availability for cost of consumption. Probabilistic SLAs specify clauses that determine availability percentile for contracted resources computed over the SLA evaluation period. **The lower the availability percentile, the cheaper the cost of resource consumption.** This type of SLA is suitable for small and medium businesses and for many enterprise grade applications.

Deterministic SLAs. (Ontic) These are, in fact, probabilistic SLAs where **resource availability percentile is 100%**. These SLAs are most stringent and difficult to guarantee. From the provider's point of view, they do not admit capacity multiplexing. Therefore this is the most costly option for service providers, which may be applied for critical services.

BITS Pilani

SLA Life Cycle

Each SLA goes through a sequence of steps starting from identification of terms and conditions, activation and monitoring of the stated terms and conditions, and eventual termination of contract once the hosting relationship ceases to exist. Such a sequence of steps is called SLA life cycle and consists of the following five phases:

1. Contract definition
2. Publishing and discovery
3. Negotiation
4. Operationalization
5. De-commissioning

SLA Life Cycle

Contract Definition: Generally, service providers define a set of service offerings and corresponding SLAs using standard templates. These service offerings form a catalog. Individual SLAs for enterprises can be derived by customizing these base SLA templates.

Publication and Discovery. Service provider advertises these base service offerings through standard publication media, and the customers should be able to locate the service provider by searching the catalog. The customers can search different competitive offerings and shortlist a few that fulfill their requirements for further negotiation.

Negotiation: Once the customer has discovered a service provider who can meet their application hosting need, the SLA terms and conditions needs to be mutually agreed upon before signing the agreement for hosting the application. For a standard packaged application which is offered as service, this phase could be automated.

For customized applications that are hosted on cloud platforms, this phase is manual. The service provider needs to analyze the application's behavior with respect to scalability and performance before agreeing on the specification of SLA. At the end of this phase, the SLA is mutually agreed by both customer and provider and is eventually signed off. SLA negotiation can utilize the WS-negotiation specification

BITS Pilani

SLA Life Cycle

Operational: SLA operation consists of SLA monitoring, SLA accounting, and SLA enforcement.

SLA monitoring involves measuring parameter values and calculating the metrics defined as a part of SLA and determining the deviations.

On identifying the deviations, the concerned parties are notified. SLA Accounting involves capturing and archiving the adherence for compliance.

As part of accounting, the application's actual performance and the performance guaranteed as a part of SLA is reported.

De-commissioning : SLA decommissioning involves termination of all activities performed under a particular when the hosting relationship between the service provider and the service consumer has ended.

SLA specifies the terms and conditions of contract termination and specifies situations under which the relationship between a service provider and a service consumer can be considered to be legally ended

SLA Life Cycle

- From the SLA perspective there are multiple challenges for provisioning the infrastructure on demand. These includes -
- The application if a black box to CSP (Cloud Service Provider) and the CSP have virtually no knowledge about the application runtime characteristics. CSP needs to determine the right amount of computing resources required for different components of application at various workloads.
- CSP needs to understand the performance bottlenecks and the scalability of the application.
- CSP analyses the application before it goes live. However, subsequent operations by customers to their applications or auto updates beside others can impact performance of the applications, thereby making applications SLA at risk.
- The risk of capacity planning is with service provider instead of customer.

SLA LC Management – Cloud Applications

SLA management of applications hosted on cloud platforms involves five phases.

1. Feasibility
2. On-boarding
3. Pre-production
4. Production
5. Termination

SLA Cloud Application LC Management

Feasibility Analysis

Managed Service Providers(MSP) conducts the feasibility study of hosting an application on their cloud platforms.

This study involves three kinds of feasibility:

- (a) **Technical feasibility,**
- (b) **Infrastructure feasibility**
- (c) **Financial feasibility.**

The technical feasibility of an application implies determining the following:

1. Ability of an application to scale out.
2. Compatibility of the application with the cloud platform being used within the MSP's data center.
3. The need and availability of a specific hardware and software required for hosting and running of the application.
4. Preliminary information about the application performance and whether they can be met by the MSP.

BITS Pilani

SLA Cloud Application LC Management

Performing **the infrastructure feasibility** involves determining the availability of infrastructural resources in sufficient quantity so that the projected demands of the application can be met.

The financial feasibility study involves determining the approximate cost to be incurred by the MSP and the price the MSP charges the customer so that the hosting activity is profitable to both of them.

A **feasibility report** consists of the results of the above three feasibility studies. The report forms the basis for further communication with the customer. Once the provider and customer agree upon the findings of the report, the outsourcing of the application hosting activity proceeds to the next phase, called “on boarding” of application. Only the basic feasibility of hosting an application has been carried in this phase. However, the detailed runtime characteristics of the application are studied as part of the on-boarding activity

SLA Cloud Application LC Management

On-boarding of Application:

Once the customer and the MSP agree in principle to host the application based on the findings of the feasibility study, the application is moved from the customer servers to the hosting platform.

Moving an application to the MSP's hosting platform is called on-boarding.

As part of the on-boarding activity, the MSP understands the application runtime characteristics using runt profilers*.

* Profiling is a method of gathering performance data in any development or deployment scenario. This is for developers and system administrators who want to gather information about application performance

SLA Cloud Application LC Management

On-boarding :

Packing of the application for deploying on physical or virtual environments. Application packaging is the process of creating deployable components on the hosting platform (could be physical or virtual). Open Virtualization Format (OVF) standard is used for packaging the application for cloud platform.

The packaged application is executed **directly on the physical servers** to capture and analyze the application performance characteristics. It allows **the functional validation of customer's application**. Besides, it provides a **baseline performance value** for the application in **non virtual environment**.

This can be used as **one of the data points for customer's performance expectation** and for **application SLA**. Additionally, it helps to identify the nature of application—that is, whether it is CPU-intensive or I/O intensive or network-intensive and the potential performance bottlenecks.

SLA Cloud Application LC Management

On-boarding :

The application is **executed on a virtualized platform** and the application performance characteristics are **noted again**. Important performance characteristics like the application's **ability to scale (out and up)** and **performance bounds** (minimum and maximum performance) are noted.

Based on the measured performance characteristics, **different possible SLAs are identified**. The **resources required** and the costs involved for each SLA are also computed.

Once the customer agrees to the set of SLAs and the cost, the MSP starts creating different policies required by the data center for automated management of the application. This implies that the management system should automatically infer the amount of system resources that should be allocated/de-allocated to/from appropriate components of the application when the load on the system increases/decreases.

SLA Cloud Application LC Management

Pre-Production

Once the determination of policies is completed as discussed in previous phase, the application is hosted in simulated production environment.

It facilitates the customer to verify and validate the MSP's findings on application's runtime characteristics agree on the defined SLA. Once both parties agree on the cost and the terms and conditions of the SLA, customer sign-off is obtained. On successful completion of this phase the MSP allows the application to go on-live.

SLA Cloud Application LC Management

Production

In this phase, the application is made accessible to its end users under the agreed SLA.

However, there could be situations when the managed application tends to behave differently in a production environment compared to the preproduction environment.

This in turn may cause sustained breach of the terms and conditions mentioned in the SLA. Additionally, customers may request the MSP for inclusion of new terms and conditions in the SLA.

If the application SLA is breached frequently or if the customer requests for a new non-agreed SLA, the on-boarding process is performed again. In the case of the former, on-boarding activity is repeated to analyze the application and its policies with respect to SLA fulfillment. In case of the latter, a new set of policies are formulated to meet the fresh terms and conditions of the SLA.

SLA Cloud Application LC Management

Termination

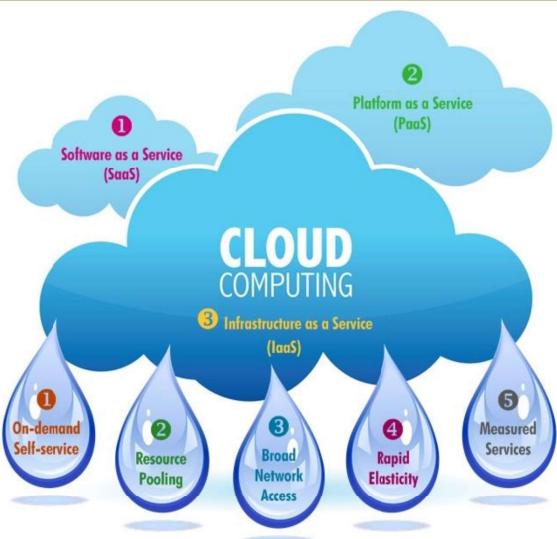
When the customer wishes to withdraw the hosted application and does not wish to continue to avail the services of the MSP for managing the hosting of its application, the termination activity is initiated.

On initiation of termination, all data related to the application are transferred to the customer and only the essential information is retained for legal compliance. This ends the hosting relationship between the two parties for that application, and the customer sign-off is obtained.

SEWI-ZG527

CLOUD COMPUTING

Session 8: 18-06-2017
This session covers DFS, GFS, MR & Hadoop



Today's Topics

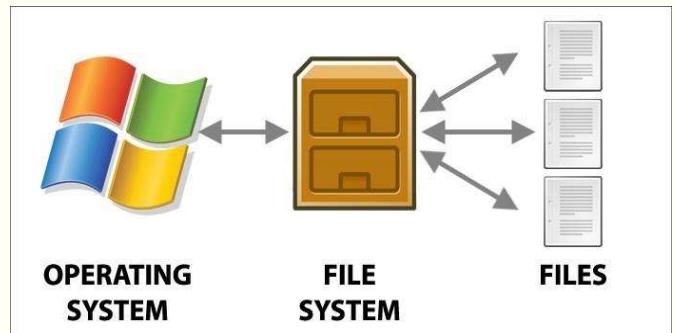


- Distributed File System
 - Introduction to DFS
 - DFS Architecture
 - Case Study: GFS
 - Case Study : HDFS
- Hadoop Components
 - Map Reduce
 - Sample Map Reduce Program
 - Map Reduce Facts
- Demo
 - Setting up a HDFS Cluster
 - Map Reduce

File System- Introduction

What is File System

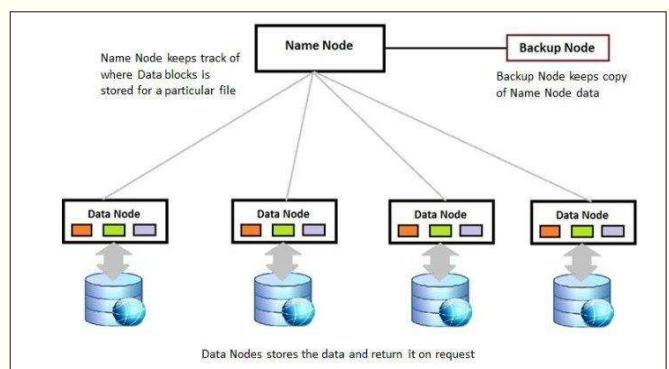
- File systems are abstraction that enable users to read, manipulate and organize data.
- Typically the data is stored in units known as files in a hierarchical tree where the nodes are known as directories.
- The file system enables a uniform view, independent of the underlying storage devices which can range between anything from floppy drives to hard drives and flash memory cards
- Since file systems evolved from stand-alone computers the connection between the logical file system and the storage device was typically a one-to-one mapping.
- Even software RAID that is used to distribute the data on multiple storage devices is typically implemented below the file system layer



Distributed File System- Introduction

What is Distributed File System

- A distributed file system is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer.
- When a user accesses a file on the server, the server sends the user a copy of the file, which is cached on the user's computer while the data is being processed and is then returned to the server.
- Ideally, a distributed file system organizes file and directory services of individual servers into a global directory in such a way that remote data access is not location-specific but is identical from any client.
- All files are accessible to all users of the global file system and organization is hierarchical and directory-based.
- Since more than one client may access the same data simultaneously, the server must have a mechanism in place (such as maintaining information about the times of access) to organize updates so that the client always receives the most current version of data and that data conflicts do not arise.



Distributed file systems typically use file or database replication (distributing copies of data on multiple servers) to protect against data access failures.

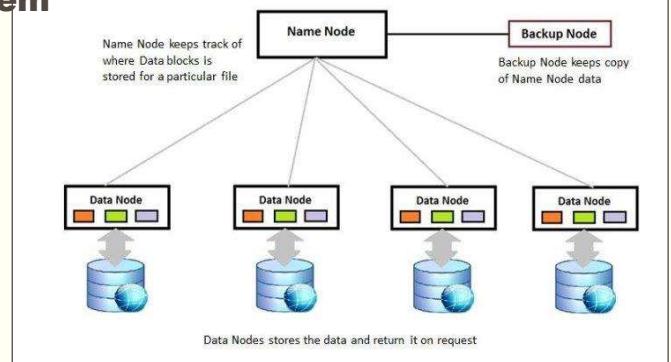
Distributed File System- Components

Components of Distributed File System

- **Service** - software entity running on one or more machines and providing a particular type of function to a priori unknown clients.
- **Server** - service software running on a single machine.
- **Client** - process that can invoke a service using a set of operations that forms its client interface.
- A client interface for a file service is formed by a set of primitive file operations (create, delete, read, write).
- Client interface of a DFS should be transparent, i.e., not distinguish between local and remote files.

Goal:

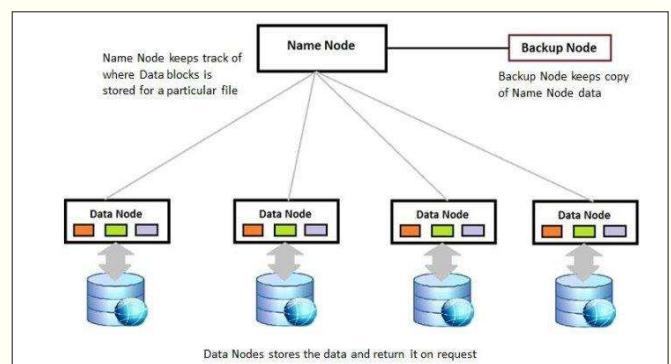
Provide common view of centralized file system provide common view of centralized file system, but distributed implementation.
Ability to open & update any file on any machine on file on any machine on network network –
All of synchronization issues and capabilities of shared local files local files

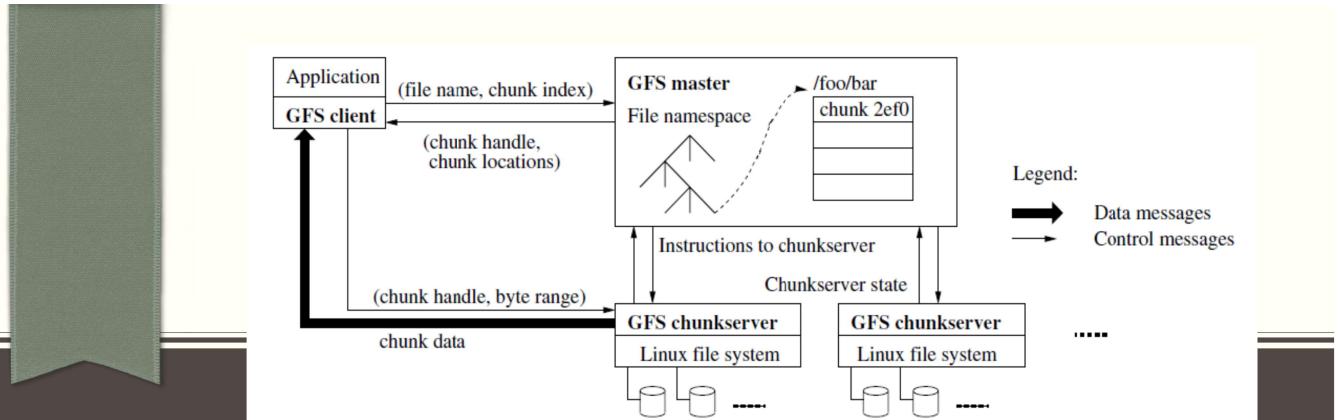


Distributed File System- Challenges

Distributed File System Challenges

- Naming and Transparency
- Remote file access
- Caching
- Stateful vs Stateless service
- Replication
- Fault Tolerance
- Security





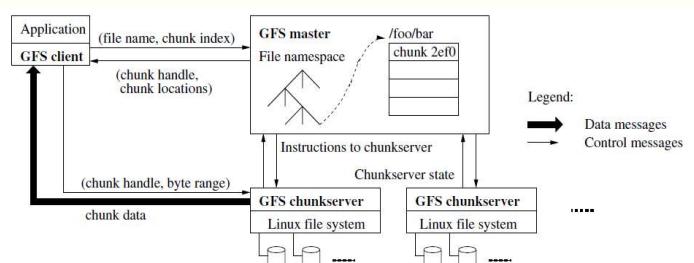
GOOGLE FILE SYSTEM

GFS

GFS

What is GFS

- GFS is a scalable distributed file system for large data intensive applications built in 2003 by Google.
- Shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability.
- The design of GFS is driven by four key observations
 - Component failures,
 - huge files,
 - mutation of files, and
 - benefits of co-designing the applications and file system API



GFS

GFS Assumptions

- GFS has high component failure rates
 - System is built from many inexpensive commodity components
- Modest number of huge files
 - A few million files, each typically 100MB or larger (Multi-GB files are common)
need to optimize for small files
- Workloads : two kinds of reads, and writes
 - Large streaming reads (1MB or more) and small random reads (a few KBs)
 - Small random reads
 - Sequential appends to files by hundreds of data producers
- High sustained throughput is more important than latency
Response time for individual read and write is not critical

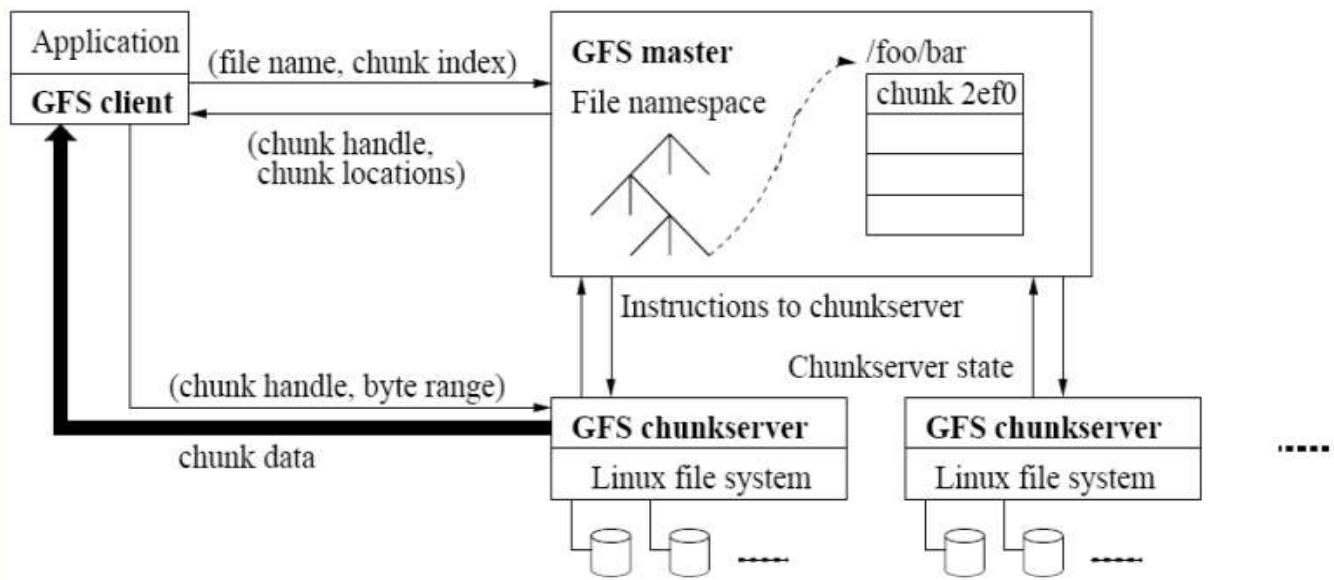
GFS

GFS is not suited

- GFS/HDFS are not a good fit for:
 - • **Low latency data access** (in the milliseconds range)
 - **Many small files!**
 - • • **Constantly changing data!**
 - • Not all details of GFS are public knowledge

GFS

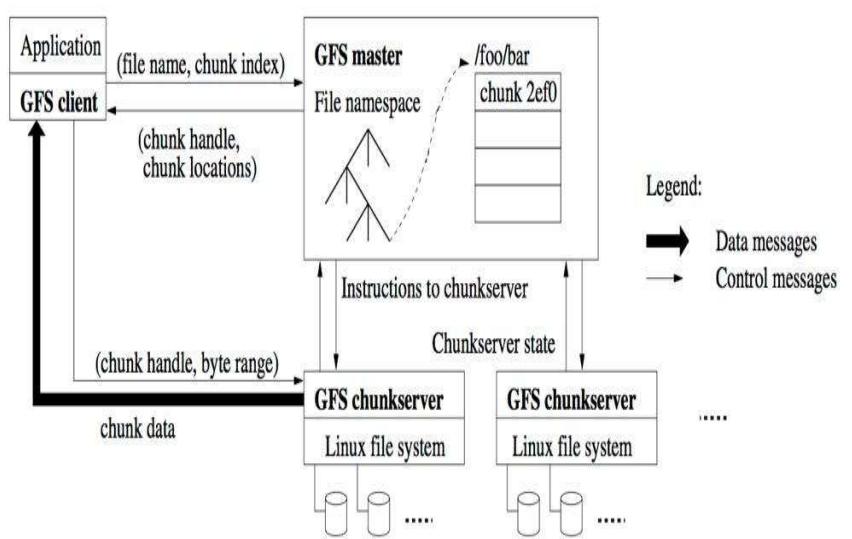
GFS Architecture



GFS

GFS Design Overview

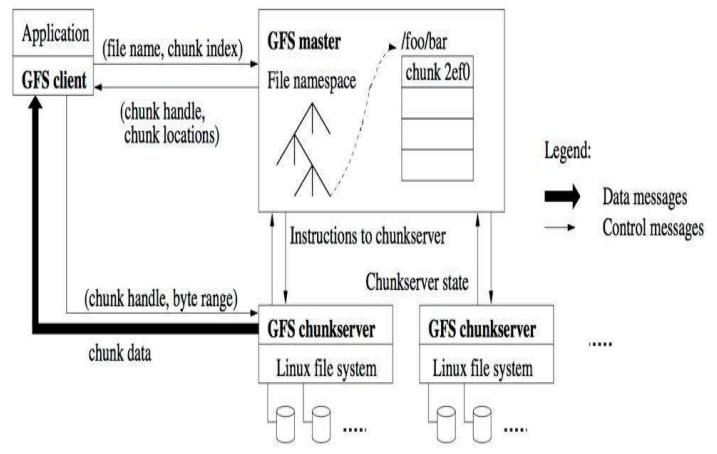
- Single Master
 - Centralized management
- Files stored as chunks
 - With a fixed size of 64MB each.
- Reliability through replication
 - Each chunk is replicated across 3 or more chunk servers
- Data caching
 - Due to large size of data sets
- Interface
 - Suitable to Google apps
 - Create, delete, open, close, read, write, snapshot, record append



GFS

Files on GFS

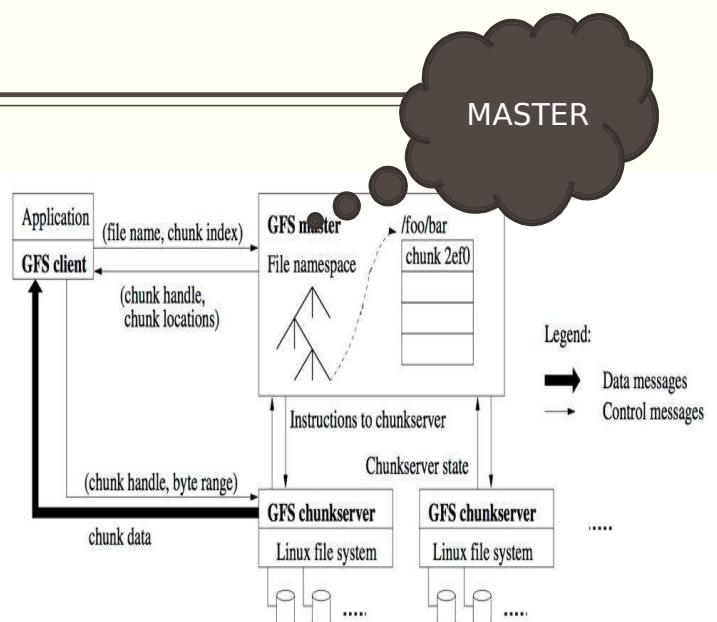
- A single file can contain many objects (e.g. Web documents)
- Files are divided into **fixed size chunks** (64MB) with unique 64 bit identifiers
- IDs assigned by GFS master at chunk creation time
- **Chunk servers** store chunks on local disk as "normal" Linux files
- Reading & writing of data specified by the **tuple** (chunk_handle, byte_range)



GFS

MASTER

- Files are **replicated** (by default 3 times) across all chunk servers
- **Master** maintains all **file system metadata!**
- Namespace, access control information, **mapping from file to chunks, chunk locations**, garbage collection of orphaned chunks, chunk migration, ...
- **Heartbeat** messages between master and chunk servers
- Is the chunk server still alive? What chunks are stored at the chunkserver?!
- **To read/write data:** client communicates with master (metadata operations) and chunk servers (data)

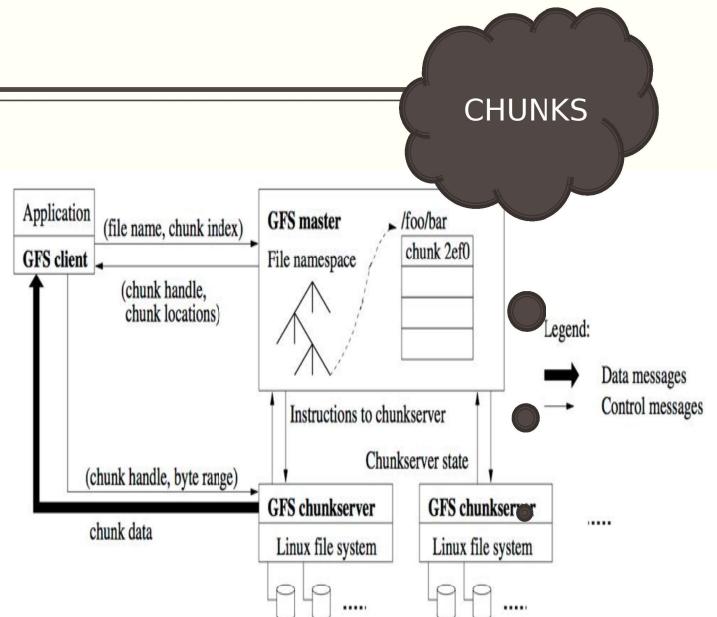


Single master in a large cluster can become a **bottleneck**
• Goal: **minimize the number of reads and writes** (thus metadata vs. data)

GFS

CHUNKS

- Chunks:
 - Fixed size of 64MB
- Advantages
 - Size of meta data is reduced
 - Involvement of Master is reduced
 - Network overhead is reduced
 - Lazy space allocation avoids internal fragmentation
- Disadvantages
 - Hot spots

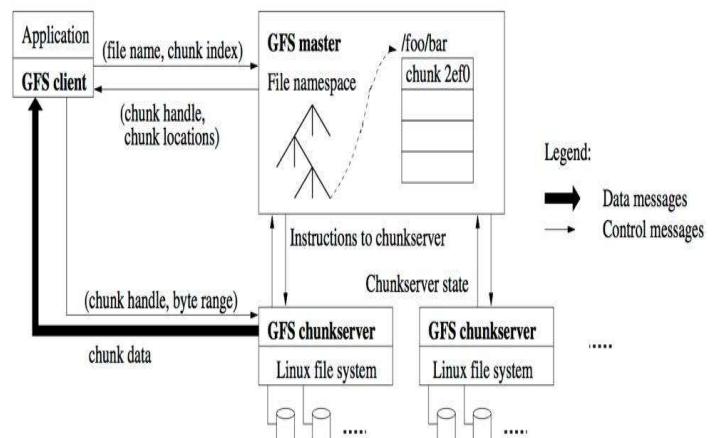


Solutions: increase the replication factor and stagger application start times; allow clients to read data from other clients

GFS

META DATA

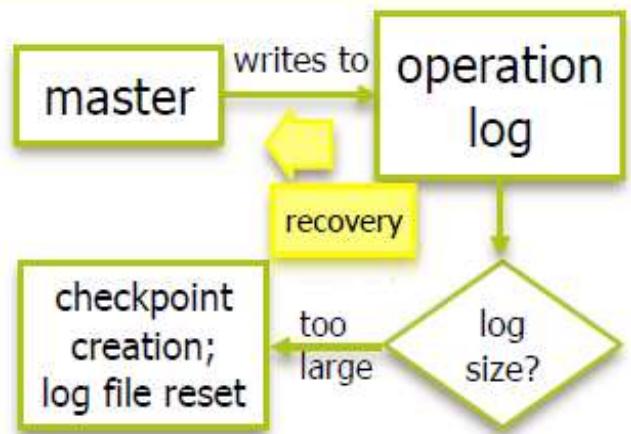
- Three major types of metadata
 - The file and chunk namespaces
 - The mapping from files to chunks
 - Locations of each chunk's replicas
- All the metadata is kept in the Master's memory
- Master “operation log”
 - Consists of namespaces and file to chunk mappings
 - Replicated on remote machines
 - 64MB chunk has 64 bytes of metadata
- Chunk locations
 - Chunk servers keep track of their chunks and relay data to Master through HeartBeat messages



GFS

OPERATIONAL LOGS

- Operation log
- Persistent record of critical metadata changes
- Critical to the recovery of the system
- Changes to metadata are only made visible to clients **after** they have been written to the operation log
 - Operation log **replicated** on multiple remote machines
 - **Before** responding to client operation, log record must have been **flashed locally and remotely**
 - Master recovers its file system from checkpoint + operation



GFS

CHUNK REPLICA

- Chunk replica placement
 - Creation of (initially empty) chunks
 - Use under utilized chunk servers; spread across racks
 - Limit number of recent creations on each chunk server
- • **Re-replication!**
 - Started once the available replicas fall below setting
 - Master instructs chunkserver to copy chunk data directly from existing valid replica
 - Number of active clone operations/bandwidth is limited
- • **Re-balancing!**
 - Changes in replica distribution for better load balancing;
 - gradual filling of new chunk servers

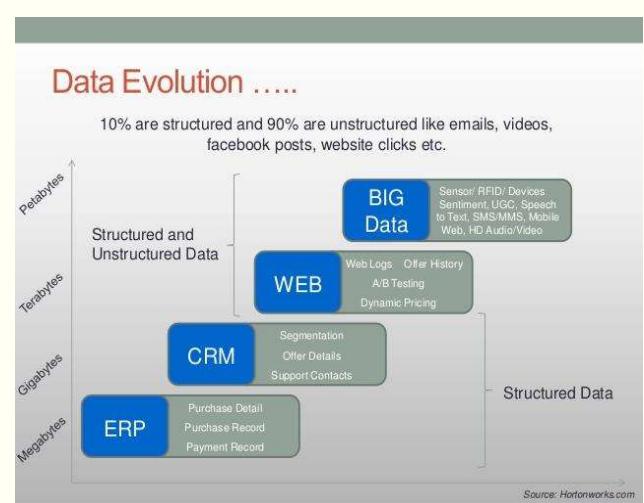
BIG DATA

HDFS



DATA Evolution

- Past decade, due to rapid progress in technology, several sources of data has emerged. Example Social networks, eCommerce, P2P networks, Climate control systems, traffic cameras etc
- The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field.
- The same amount was created in every two days in 2011, and in every ten minutes in 2013. Today we create 2.5 quintillion bytes of data per day.
- However, 90% of the data was created in the past 2 – 5 years only !!!
- Though all this information produced is meaningful and can be useful when processed, it is being neglected



DATA Evolution

What is BIG DATA

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

We are in a knowledge economy.

Data is an important asset to any organization

Discovery of knowledge; Enabling discovery;
annotation of data

Complex computational models

No single environment is good enough: need elastic,
on-demand capacities

We are looking at newer

Programming models, and

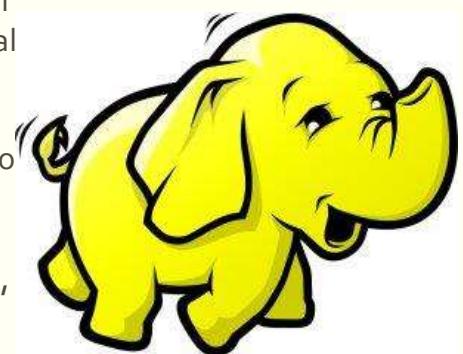
Supporting algorithms and data structures.



Hadoop

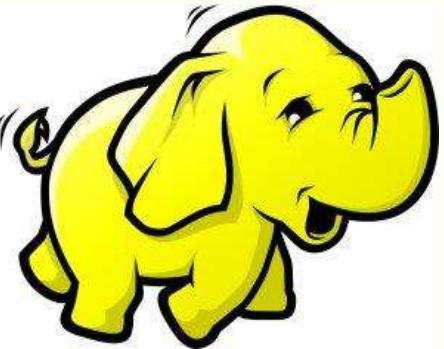
• HADOOP

- Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005.
- Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.
- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes.
- In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.



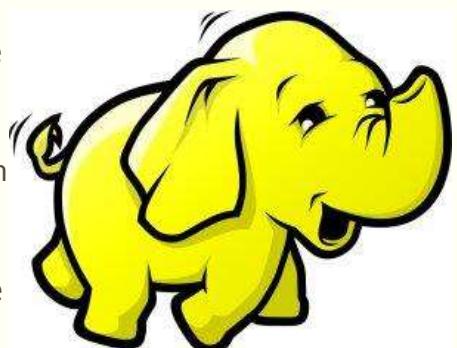
Hadoop – HDFS Design

- HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.
- Very large files “Very large” in this context means files that hundreds of megabytes, gigabytes, or terabytes in size. There are Hadoop clusters running today that store petabytes of data.
- Streaming data access HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern.
- A dataset is typically generated or copied from source, and then various analyses are performed on that dataset over time.
- Each analysis will involve a large proportion, if not all, of the dataset, so the time to read the whole dataset is more important than the latency in reading the first record.



Hadoop – HDFS Design

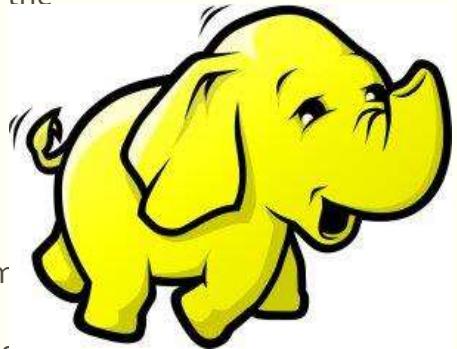
- Commodity hardware Hadoop doesn't require expensive, highly reliable hardware.
- It's designed to run on clusters of commodity hardware (commonly available hardware that can be obtained from multiple vendors) for which the chance of node failure across the cluster is high, at least for large clusters.
- HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure.



Hadoop – HDFS Design

HDFS is not Recommended when:

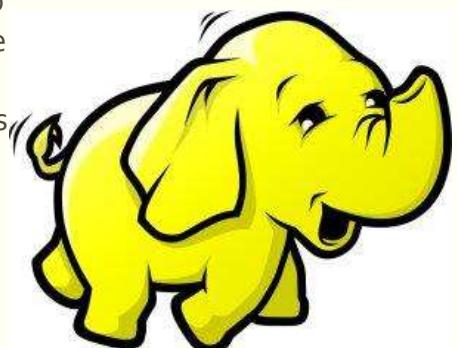
- **Low-latency data access**
 - Applications that require low-latency access to data, in the tens of milliseconds range, will not work well with HDFS.
 - Remember, HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency.
- **Lots of small files**
 - Because the namenode holds filesystem metadata in memory, the limit to the number of files in a filesystem governed by the amount of memory on the namenode.
 - As a rule of thumb, each file, directory, and block takes about 150 bytes. So, for example, if you had one million files, each taking one block, you would need at least 300 MB of memory.
 - Although storing millions of files is feasible, billions is beyond the capability of current hardware.



Hadoop – HDFS Design

HDFS is not Recommended when:

- **Multiple writers,**
 - Arbitrary file modifications Files in HDFS may be written to by a single writer. Writes are always made at the end of the file, in append-only fashion.
 - There is no support for multiple writers or for modifications at arbitrary offsets in the file.



Hadoop – HDFS Design

HDFS BLOCKS

- A disk has a block size, which is the minimum amount of data that it can read or write.
- File systems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size.
- File system blocks are typically a few kilobytes in size, whereas disk blocks are normally 512 bytes.
- This is generally transparent to the file system user who is simply reading or writing a file of whatever length.
- HDFS, too, has the concept of a block, but it is a much larger unit 128 MB by default.
- Like in a file system for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a file system for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage. (For example, a 1 MB file stored with a block size of 128 MB uses 1 MB of disk space, not 128 MB.)

Note:

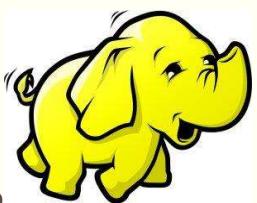
#Blocks = File Size/ Block size

#Chunks = File Size/ Chunk Size

Hadoop – HDFS Design

HDFS BLOCKS

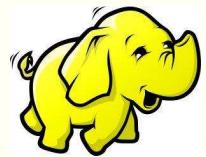
- **Why Is a Block in HDFS So Large?** HDFS blocks are large **compared to disk blocks**, and the reason is to **minimize the cost of seeks**.
- If the block is large enough, the time it takes to transfer the data from the disk can be significantly longer than the time to seek to the start of the block.
- Thus, transferring a large file made of multiple blocks operates at the disk's transfer rate.
- The default is actually 128 MB, although many HDFS installations use larger block sizes.
- This figure will continue to be revised upward as transfer speeds grow with new generations of disk drives.



Hadoop – HDFS Design

HDFS NAME NODES & DATA NODES

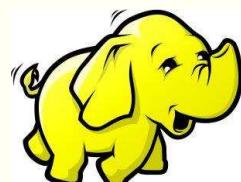
- An HDFS cluster has two types of nodes operating in a **master – worker pattern**: a namenode (the master) and a number of datanodes (workers).
- The namenode manages the filesystem namespace.
- It maintains the filesystem tree and the metadata for all the files and directories in the tree.
- This information is stored persistently on the local disk in the form of two files:
 - the namespace image and
 - the edit log.
- The **namenode** also knows the **datanodes** on which all the blocks for a given file are located; however, it does not store block locations persistently, because this information is **reconstructed from datanodes** when the system starts.



Hadoop – HDFS Design

HDFS NAME NODES & DATA NODES

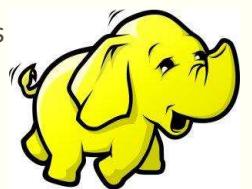
- A client accesses the file system on behalf of the user by communicating with the **namenode** and datanodes.
- The client presents a file system interface similar to a Portable Operating System Interface (POSIX), so the user code does not need to know about the namenode and datanodes to function.
- Datanodes are the workhorses of the filesystem.
- Datanodes store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.
- Without the namenode, the filesystem cannot be used. In fact, if the machine running the namenode were obliterated, all the files on the filesystem would be lost since there would be no way of knowing how to reconstruct the files from the blocks on the datanodes. For this reason, it is important to make the namenode resilient to failure, and Hadoop provides two mechanisms for this.



Hadoop – HDFS Design

HDFS REDUNDANCY (High Availability)

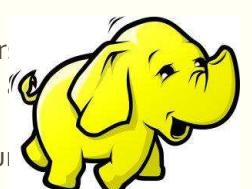
- The first way is to back up the files that make up the persistent state of the file system metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems.
- These writes are synchronous and atomic. The usual configuration choice is to write to local disk as well as a remote NFS mount.
- It is also possible to run a secondary namenode, which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image
- The secondary namenode usually runs on a separate physical machine because it requires plenty of CPU and as much memory as the namenode to perform the merge.
- It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing.
- However, the state of the secondary namenode lags that of the primary, so in the event of total failure of the primary, data loss is almost certain. The usual course of action in this case is to copy the namenode's metadata files that are on NFS to the secondary and run it as the new primary.



Hadoop – HDFS Design

HDFS BLOCK CACHING

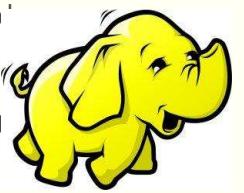
- Normally a datanode reads blocks from disk, but for frequently accessed files the blocks may be explicitly cached in the datanode's memory, in an off-heap block cache.
- By default, a block is cached in only one datanode's memory, although the number is configurable on a per-file basis.
- Job schedulers (for MapReduce, Spark, and other frameworks) can take advantage of cached blocks by running tasks on the datanode where a block is cached, for increased read performance.
- A small lookup table used in a join is a good candidate for caching, for example. User applications instruct the namenode which files to cache (and for how long) by adding a cache directive to a cache pool.
- Cache pools are an administrative grouping for managing cache permissions and resource usage.



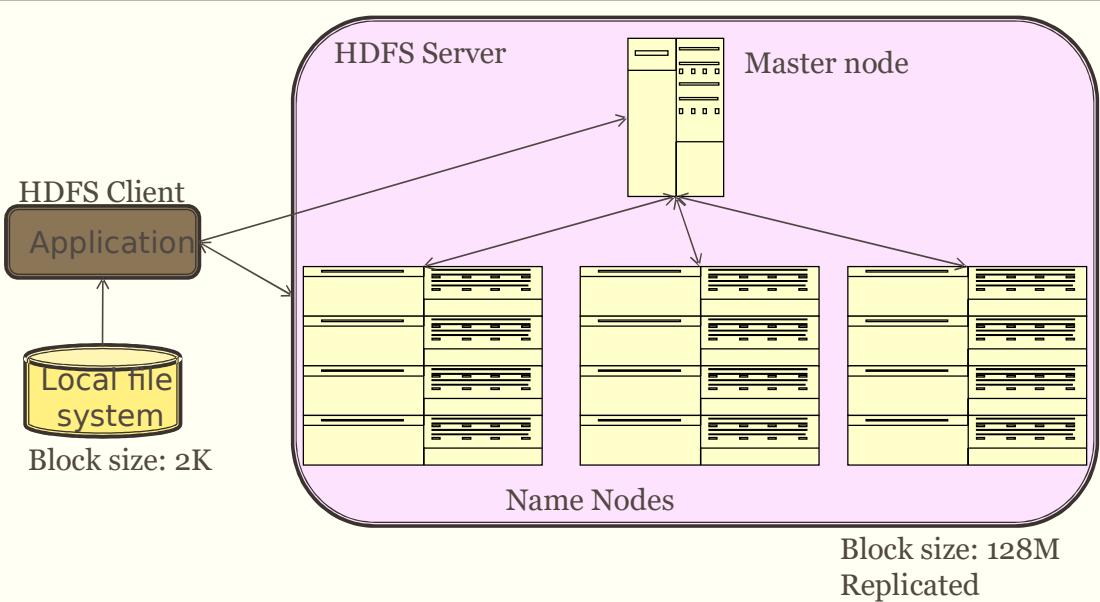
Hadoop – HDFS Design

HDFS FEDERATION

- The namenode keeps a reference to every file and block in the filesystem in memory, which means that on very large clusters with many files, memory becomes the limiting factor for scaling
- HDFS federation, introduced in the 2.x release series, allows a cluster to scale by adding namenodes, each of which manages a portion of the filesystem namespace.
- For example, one namenode might manage all the files rooted under /user, say, and a second namenode might handle files under /share.
- Under federation, each namenode manages a namespace volume, which is made up of the metadata for the namespace, and a block pool containing all the blocks for the files in the namespace.
- Namespace volumes are independent of each other, which means namenodes do not communicate with one another, and furthermore the failure of one namenode does not affect the availability of the namespaces managed by other name nodes.

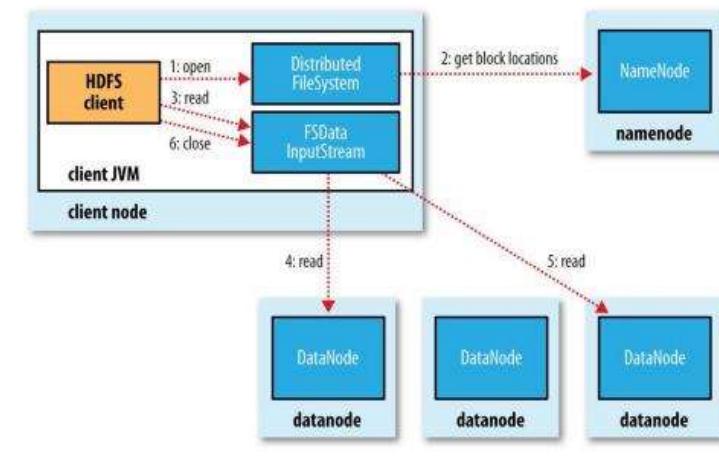


Hadoop Distributed File System



Hadoop – HDFS Design

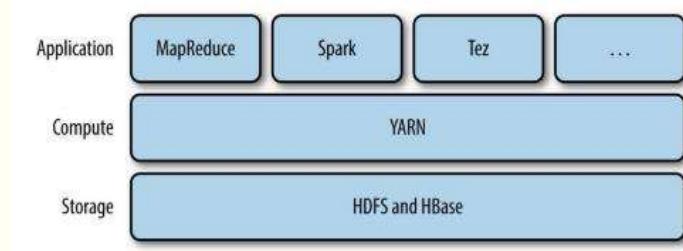
HDFS File Read



1. The client opens the file it wishes to read by calling `open()` on the `FileSystem` object.
2. `DistributedFileSystem` calls the namenode, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file. For each block, the namenode returns the addresses of the datanodes that have a copy of that block.
3. The `DistributedFileSystem` returns an `FSDataInputStream` (an input stream that supports file seeks) to the client for it to read data from. `FSDataInputStream` in turn wraps a `DFSInputStream`, which manages the datanode and namenode I/O. The client then calls `read()` on the stream.
- 4 & 5. When the first block end is reached the `DFSInputStream` will automatically search for the next node and close previous connections.

Hadoop – HDFS Design

YARN



Apache YARN (Yet Another Resource Negotiator) is Hadoop's cluster resource management system.

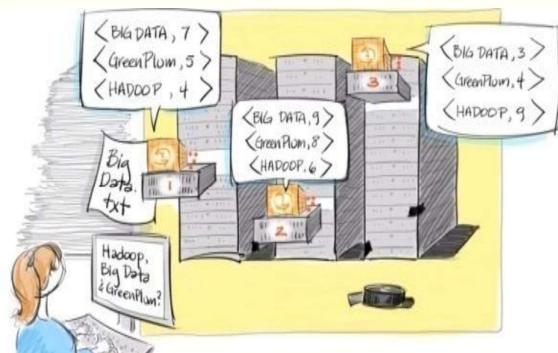
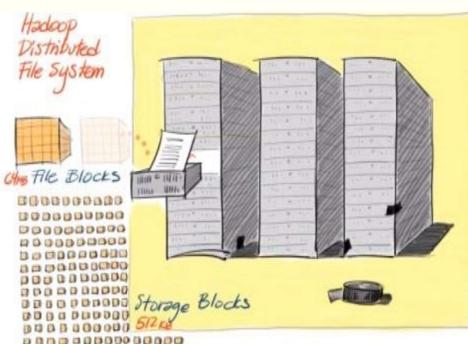
YARN was introduced in Hadoop 2 to improve the MapReduce implementation, but it is general enough to support other distributed computing paradigms as well.

YARN provides APIs for requesting and working with cluster resources, but these APIs are not typically used directly by user code. Instead, users write to higher-level APIs provided by distributed computing frameworks, which themselves are built on YARN and hide the resource management details from the user.

MapReduce (Data Processing Framework)

MapReduce

Software Framework for easily running applications	Processes large amount of data in parallel	Using large clusters having thousands of nodes	Nodes of commodity hardware	In a reliable and fault-tolerant manner
--	--	--	-----------------------------	---



BITS Pilani

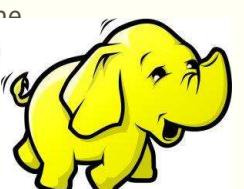
Hadoop – HDFS MAP REDUCE

HDFS MR

MapReduce program executes in three stages, namely **map stage**, **shuffle stage**, and **reduce stage**.

Map stage : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage : This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.



During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

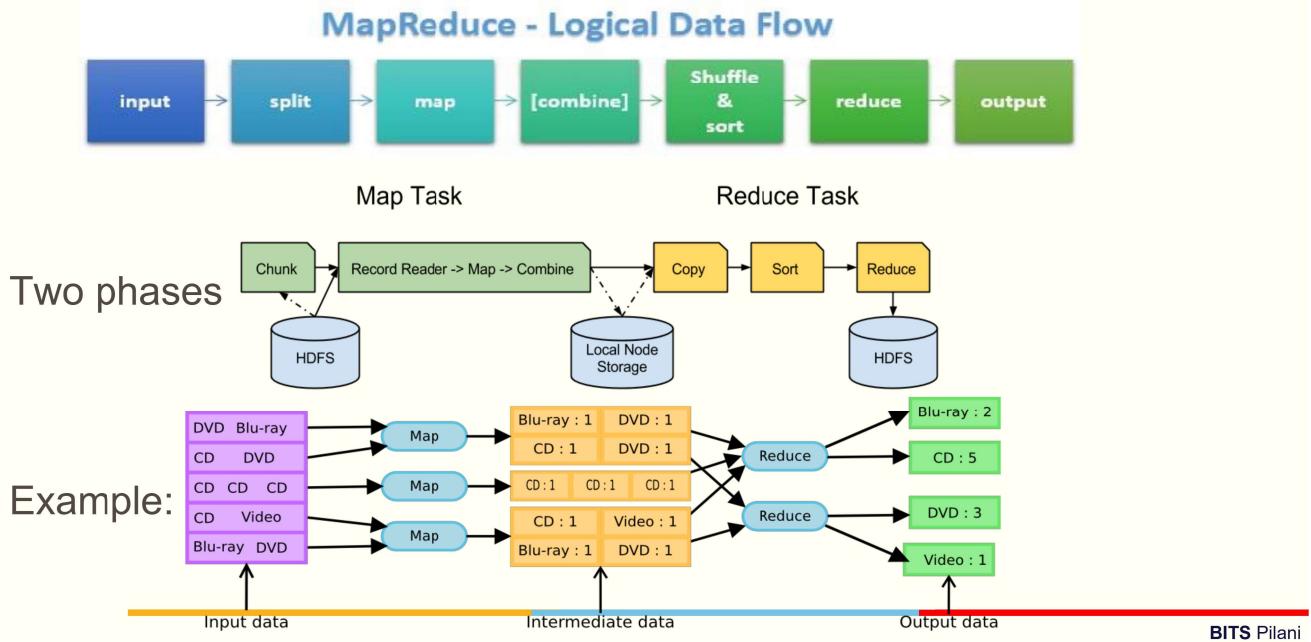
The framework manages all the details of data-passing such as **issuing tasks**, **verifying task completion**, and copying data around the cluster between the nodes.

Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

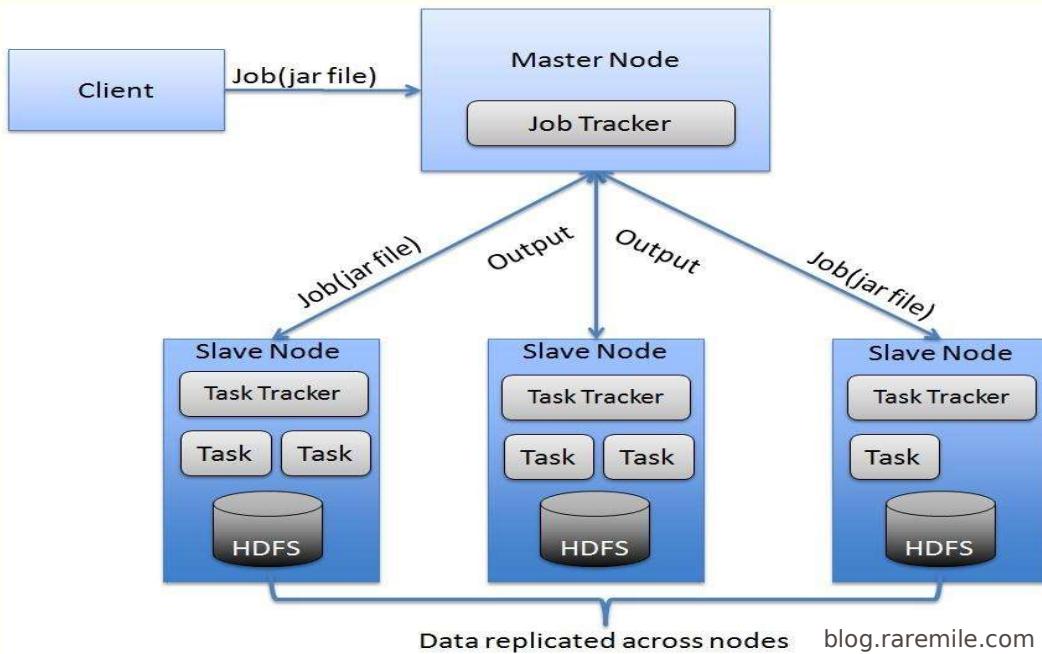
After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

MapReduce Processing flow

Logical flow:



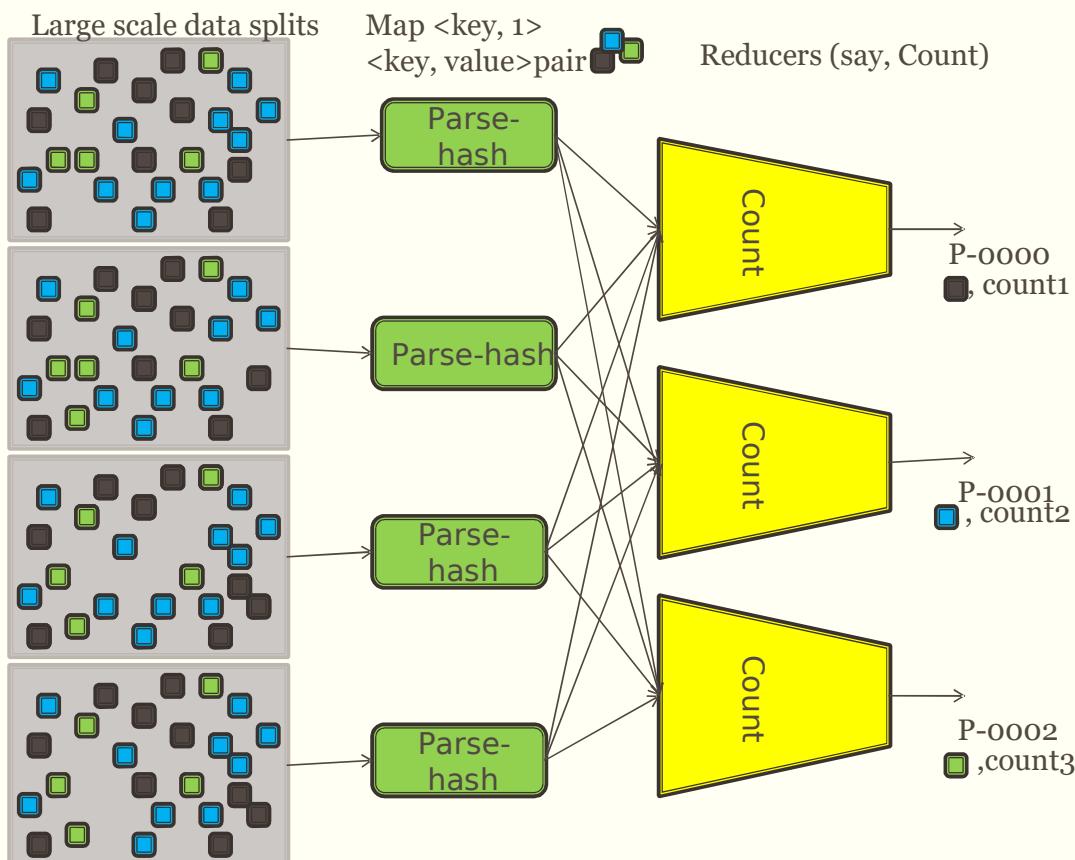
Architecture Overview



Classes of problems “mapreducible”

- Benchmark for comparing: Jim Gray’s challenge on data-intensive computing.
Ex: “Sort”
- Google uses it for wordcount, adwords, pagerank, indexing data.
- Simple algorithms such as grep, text-indexing, reverse indexing
- Bayesian classification: data mining domain
- Facebook uses it for various operations: demographics
- Financial services use it for analytics
- Astronomy: Gaussian analysis for locating extra-terrestrial objects.
- Expected to play a critical role in semantic web and in web 3.0

41



Fault Tolerance in MapReduce

1. If a task crashes:

- Retry on another node
 - OK for a map because it had no dependencies
 - OK for reduce because map outputs are on disk
- If the same task repeatedly fails, fail the job or ignore that input block

2. If a node crashes:

- Relaunch its current tasks on other nodes
- Relaunch any maps the node previously ran
 - Necessary because their output files were lost along with the crashed node

BITS Pilani

Fault Tolerance in MapReduce

3. If a task is going slowly (straggler):

- Launch second copy of task on another node
- Take the output of whichever copy finishes first, and kill the other one
 - Critical for performance in large clusters

Some Facts about MR

Hadoop requires numerous large data centers. The top five U.S. Internet firms operate over 300 data centers in the continental U.S., which cost around \$1 billion each to construct and consume over 200,000 square feet each. Each data center is two to three times larger than a Wal-Mart. The U.S. government operates over 2,000 data centers.

Hadoop data centers employ few workers. Each \$1 billion Hadoop data center employs fewer than 50 people, which is 10 times less than a single Wal-Mart Super Center. Their operation doesn't contribute to local economies. Internet firms are offered tax free status, municipal bonds, and little or no-cost real estate for their data centers.

Hadoop consumes a lot of U.S. energy. Hadoop clusters consume 2.2% of America's power (i.e., 567 Terawatts out of 25,776 Terawatts generated each year in the U.S.). U.S. nuclear power plants generate 400 Terawatts of power each year. Therefore, Hadoop clusters consume all power generated by U.S. nuclear power plants (and then some).

Hadoop is not energy efficient. Hadoop clusters consume a lot of power even when idle. Power consumption starts at around 10 kilowatts, which is close to the idle power of the nodes in the cluster (for a small one). Ten Petabytes of raw storage takes four times that amount (i.e., 40 kilowatts of power during idle state). A single Hadoop data center consumes about 36 megawatts per year (requiring highly-specialized power, space, and cooling).

BITS Pilani

Some Facts about MR

Hadoop is capital intensive and creates bottlenecks. Hadoop clusters cost about \$10 million. This creates problems for purchasing and engineering. Most firms don't have \$10 million just lying around. Larger organizations purchase hardware in bigger increments. This creates bottlenecks for lean and agile methods, which complete requirements in a few hours, days, and weeks. Furthermore, Hadoop clusters require specialized power, space, and cooling, and initial clusters may be incompatible with data centers. It may be necessary to purchase multiple clusters before solutions are found, which increases costs and risks (and hinders engineering work flow).

Hadoop doesn't scale in a linear fashion. Academia hosts an annual contest to see which computer system can sort a terabyte of data the fastest. Yahoo has designed a 1,000 node cluster just to compete in this contest. Yahoo generally wins this contest every year using simple, low-level Assembly language-like Hadoop programs. Simple tests like this show that Hadoop scales in a linear fashion (i.e., adding another node does not incur any additional operating system overhead in its "shared-nothing" paradigm). However, practical industrial-strength Hadoop applications create thousands if not millions of parallel operations during the shuffle and sort phase, which saturates the cluster's networking switch. Oh yeah, the network is the only shared resource in this shared-nothing paradigm.

Some Facts about MR

Hadoop violates your privacy. Hadoop is a write-once, read-many times paradigm. Once information is written, it is stored forever. It is the basis for blockbuster Internet applications like Facebook, Yahoo mail, Google mail, Amazon EC2, Microsoft's Azure, Ebay, and many more, which have hundreds of millions of users each. Once a Gmail message is sent, it is saved forever. Users delete email messages, but they are merely quarantined from further use (not deleted). All of those sweet-nothings you whisper to your significant other are saved forever in Twitter, Gmail, and Facebook. Government agencies make thousands of requests for your sweet nothings each year. Law enforcement agencies monitor social networks of international travelers and act upon misunderstood colloquialisms.

Hadoop's complexity is understood by few. Hadoop is a complex, byte-level parallel processing computer programming language. It is akin to low-level operating system file manipulation in the Assembly or C programming languages. Technologically speaking, Hadoop takes us back to the 1950s and 1960s in terms of operating systems, programming, and database processing in order to manipulate the Petabytes of data produced by 3 billion Internet users each day. Only a few human calculator-like computer programmers can really master it. Fewer than 10% of Facebook's programmers have mastered Hadoop, and Facebook is the largest consumer of Hadoop technologies.

BITS Pilani

Session Agenda



Cloud Computing – Engagements in the Cloud

- Serverless Computing
- Backend as a Service (BaaS)
- Mobile BaaS (MBaaS)
- Function as a Service (FaaS)

Future Directions in Cloud

- After Migration, What Next
- Evolution of EDGE computing
- Multi-clouds, a de facto standard

Course Wrap Up

- Review BEL & AR & IR Leases
- Exam Pattern discussion



Cloud Deployments

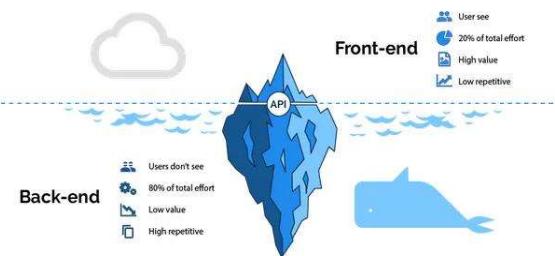


Perception

Behind every **web or mobile application**, there is an array of **backend services**.

They **support** the applications the **frontend consumers** are using and seeing every day. However, the amount of **work required** to create and manage this **backend** **is never simple and straightforward**.

At the same time, most of the **business organizations** want to **save themselves the money and time** required to redevelop the wheel **from scratch every time**. That's why they are preferring to go with an effective **Backend as a Service** solution to get the things done effectively.



BaaS Defined



A platform that

- automates backend side development
- takes care of the cloud infrastructure



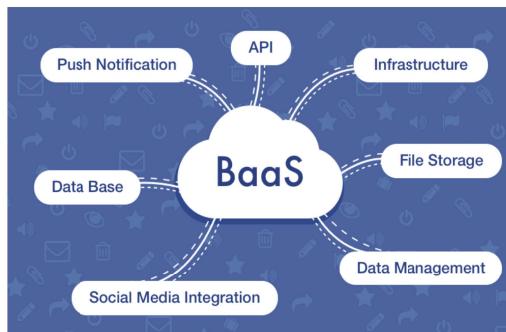
App teams

- outsource the responsibilities of running and maintaining servers to a third party
- focus on the frontend or client-side development



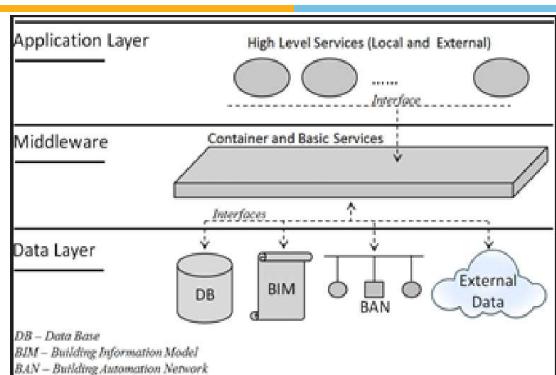
Provides a set of tools to help developers to create a backend code speedily with help of ready to use features.

BaaS Capabilities



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

BaaS Architecture



The **third layer** connects the application servers to the Internet, and it's composed of **load balancers and CDNs**.

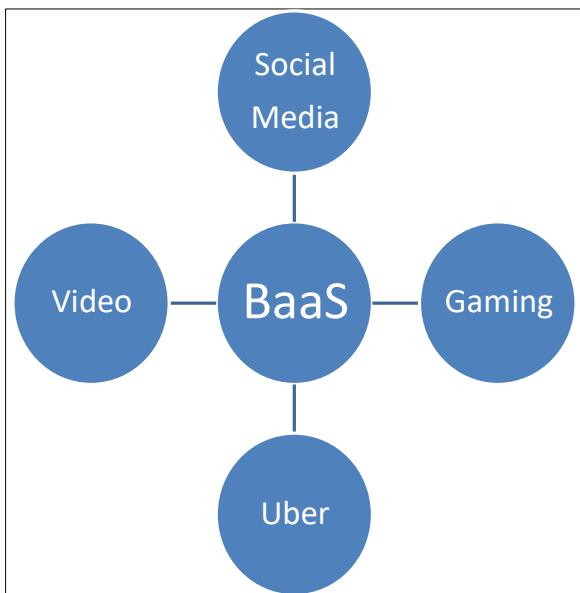
The **first layer** is the foundation and contains the database servers.

A **database cluster** will have at least two servers to replicate data and a backup routine to retrieve data. Most BaaS providers use NoSQL databases on their technology stacks due to scaling flexibility, but there is a growing trend to use SQL databases like Postgres.

The **second layer** is the **application cluster** and contains multiple servers to process requests. The quantity of servers fluctuates throughout the time of the day, and auto-scaling procedures are necessary to fulfill the group with the correct amount of servers.

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

BaaS Examples



Social Media

Login Panels, multi varied data formats to store & process, user authentication

Uber

Push notification, location based services, user authentication

Video Streaming

Get watchlist, curated play list creation, subscription services

Gaming Platform

User authentication, database management, logs

BaaS vs IaaS

Imagine you would like to **build a new software project** and that you will **not use a BaaS**. The first step before you start developing the backend side code is to **set up the servers**. Here is how it will work:

- Login on AWS or any other cloud.
- Go to Instances
- Launch Instance
- Select the Operating System
- Instance Size, Type
- Configure Instance Details
 - Number of instances ,Network, IP, Monitoring, Other settings like Auto Scaling, IAM, etc
- Add Storage
- Security Settings



All right, your **instance is up and running**, and now you can start coding! **Not really!** That is only **the first step** of the process, and you will still need to **install the web-server, database, framework, etc.**

After all that is done, you can start coding. The time to perform this process can range from **a few hours** (for a small project with skilled backend developers) to more than **a day for large environments**.

This same process using a **backend as a service** will be done with a few clicks and take no more than a few minutes.

BaaS Pro & Con

Advantages

- Speedy Development
- Reduced Development price
- Serverless, and no need to manage infrastructure



Disadvantages

- Less flexible as compared to custom coding / deployments
- Less customization in comparison to a custom backend
- Vendor lock-in possible

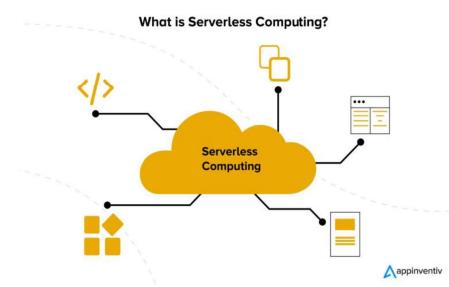
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956



BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Serverless Computing



Serverless Computing

Serverless computing is a method of providing **backend services on an as-needed basis**.

A **serverless provider** allows **users** to write and **deploy code** without the hassle of worrying about the **underlying infrastructure**.

A company that gets backend services from a serverless vendor is **charged** based on **their computation** and do not have to **reserve and pay for a fixed amount of bandwidth or number of servers**, as the service is auto-scaling.



Note that despite the name serverless, physical servers are still used but developers do not need to be aware of them.

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

BaaS vs Serverless

There is some overlap between **BaaS and serverless computing**, because in both the developer only has to write their application code and doesn't think about the backend. In addition, many BaaS providers also offer serverless computing services. However, there are significant operational differences between applications built using BaaS and a true serverless architecture.

How the application is constructed

The backends of **serverless applications** are broken up into **functions**, each of which responds to events and performs one action only. BaaS server-side functionalities, meanwhile, are **constructed however the provider wants**, and developers **don't have to concern** themselves with coding anything other than the frontend of the application.

When code runs

Serverless architectures are **event-driven**, meaning they run in response to events. Each function only runs when it is triggered by **a certain event**, and it does not run otherwise. Applications built with BaaS are usually not event-driven, meaning that they **require more server resources**.

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



BaaS vs Serverless

Where code runs

Serverless functions can be run from anywhere on any machine, as long as they are still in communication with the rest of the application, which makes it possible to incorporate edge computing into the application's architecture by running code at the network's edge. BaaS is not necessarily set up to run code from anywhere, at any time (although it can be, depending on the provider).

How the application scales

Scalability is one of the biggest differentiators separating serverless architectures from other kinds of architecture. In serverless computing, the application automatically scales up as usage increases. The cloud vendor's infrastructure starts up ephemeral instances of each function as necessary. BaaS applications are not set up to scale in this way unless the BaaS provider also offers serverless computing and the developer builds this into their application.

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



Serverless on AWS

Serverless is a way to describe the services, practices, and strategies that enables building more agile applications to foster innovations and responses to the changes faster

With serverless computing, infrastructure management tasks like capacity provisioning and patching are handled by AWS, developer can focus on only writing code that serves customers

Serverless services like AWS Lambda come with automatic scaling built-in high availability and a pay-for-value billing model



Lambda is an event-driven compute service that enables to run code in response to events from over 150 natively-integrated AWS and SaaS sources all without managing any servers

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

Serverless on AWS

 AWS Lambda <p>Run code without provisioning or managing servers and pay only for the resources you consume</p>	 Amazon Fargate <p>Run serverless containers on Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS)</p>
 Amazon EventBridge <p>Build an event-driven architecture that connects application data from your own apps, SaaS, and AWS services</p>	 AWS Step Functions <p>Coordinate multiple AWS services into serverless workflows so you can build and update apps quickly</p>
 Amazon SQS <p>Decouple and scale microservices with message queues that send, store, and receive messages at any volume</p>	 AWS AppSync <p>Create a flexible API to securely access, manipulate, and combine data from one or more data sources</p>
 Amazon SNS <p>Get reliable high throughput pub/sub, SMS, email, and mobile push notifications</p>	 Amazon API Gateway <p>Create, publish, maintain, monitor, and secure APIs at any scale for serverless workloads and web applications</p>

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

Serverless Advantage



1. No server management

Serverless computing does not mean that there are no servers, but it definitely means that developers or companies do not have to worry about these servers. These servers are managed by vendors and developers can focus on expanding their application without being worried about their server capacity.



2. Lower cost

This is, obviously, very cost effective because a developer is only paying for the server space they are using.



3. Flexible scalability

This is one of the important advantages of serverless computing. A developer does not have to worry about scalability of their web application because serverless computing has the ability to scale according to traffic volumes.



4. Fewer things to worry

A company or developer does not have to worry about security, patch, bugs and many other things because their servers are managed by vendors.

Serverless Disadvantages



1. Not suitable for long term tasks:

Serverless are a good option for short term or real time tasks. If a task takes longer time to complete then the company might end up paying more for compute time. For example, if a large file has to be uploaded that takes more time then it will require additional functions till it's complete and the developer ends up paying more for that.



2. High non-performance penalty:

A business pays for what they use, but if they end up not using these functions then there will be a high non-performance penalty. These functions may suffer from cold start penalty as well and can be very slow when used after a certain period of time.

3. Vendors lock in:

When a business depends on a vendor for all its backend services for a web application, it ends up losing control over their hardware, updates and run times. If they want to switch then it gets very difficult. A business or developer has to re-engineer if they wish to switch to another service provider.



4. Security concerns:

New security concerns are introduced when a vendor provides servers to a business. This can be a huge problem if the application contains personal or sensitive information like credit card details. This happens because companies are not given their own physical servers, vendors will be running code for many customers on a single server.

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

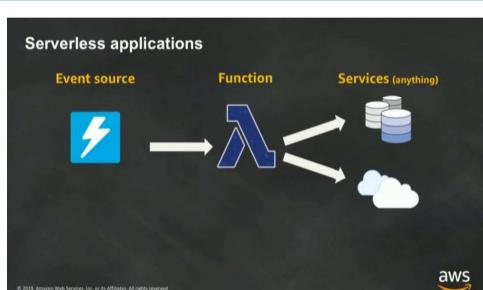


BITS Pilani

Pilani | Dubai | Goa | Hyderabad



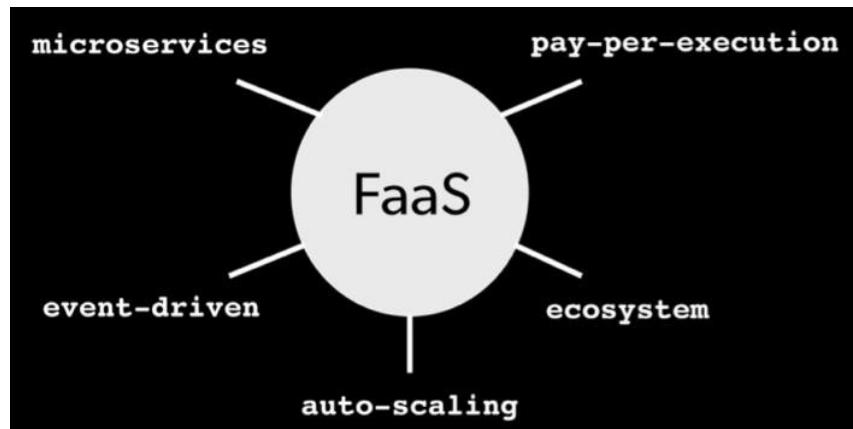
Function as a Service





FaaS

Function-as-a-Service (FaaS) is a serverless way to execute modular pieces of code on the edge. FaaS lets developers write and update a piece of code on the fly, which can then be executed in response to an event, such as a user clicking on an element in a web application. This makes it easy to scale code and is a cost-efficient way to implement microservices.



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

Microservice

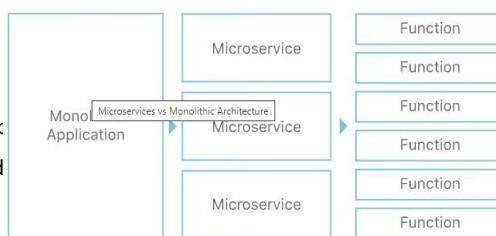
The approach of building an application out of a set of modular components is known as microservice architecture.

Dividing an application into microservices is appealing to developers because it means they can create and modify small pieces of code which can be easily implemented into their codebases.

This is in contrast to monolithic architecture, in which all the code is interwoven into one large system.

With large monolithic systems, even a minor change to the application requires a hefty deploy process.

FaaS eliminates this deploy complexity. Using serverless code like FaaS, web developers can focus on writing application code, while the serverless provider takes care of server allocation and backend services.



FaaS



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

FaaS Benefits

Focus more on code, not infrastructure:

With FaaS, can divide the server into functions that can be scaled automatically and independently so don't have to manage infrastructure

This allows to focus on the app code and can dramatically reduce time-to-market

Pay only for the resources you use, when you use them:

With FaaS, you pay only when an action occurs

When the action is done, everything stops—no code runs, no server idles, no costs are incurred

FaaS is, therefore, cost-effective, especially for dynamic workloads or scheduled tasks

FaaS also offers a superior total-cost-of-ownership for high-load scenarios

Scale up or down automatically:

With FaaS, functions are scaled automatically, independently, and instantaneously, as needed

When demand drops, FaaS automatically scales back down

Get all the benefits of robust cloud infrastructure:

FaaS offers inherent high availability because it is spread across multiple availability zones per geographic region

can be deployed across any number of regions without incremental costs

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



BITS Pilani

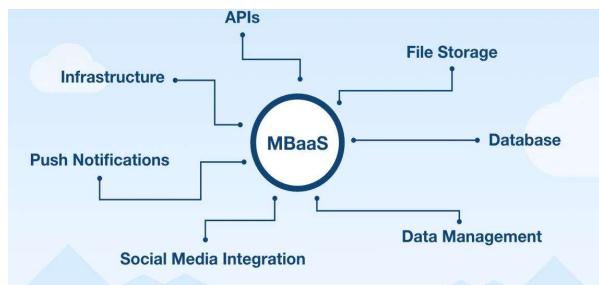
Pilani | Dubai | Goa | Hyderabad

Mobile BaaS



MBaaS

mBaaS – Mobile backend as a service helps developers in linking their applications, either mobile or websites, to the cloud via application programming interfaces (API). Other than this, the mobile backend as a service is also making grounds for helping developers in accelerating backend development, improve user management, provides push notifications, and much more. Common mBaaS features include database graphical interface, APIs, email verification, reset password, push-notifications, and more.



What are the core features of a mBaaS?

- Database
- Storage
- APIs
- Notifications
- Authentication

MBaaS Architecture

The first layer - Database

Is the foundation and contains the database servers

A database cluster has at least two servers to replicate data and a backup routine to retrieve data

The second layer - Application

Is the application cluster and contains multiple servers to process requests

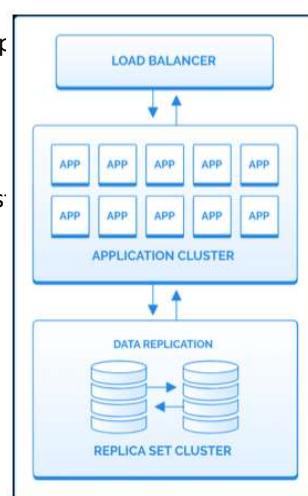
Quantity of servers fluctuates throughout the time of the day

Auto-scaling procedures are necessary to fulfill the correct number of servers

The third layer - Gateway

Connects the application servers to the Internet

Composed of load balancers and CDNs





Future Directions

What Next in Cloud

2020 VS 2025

Popular Computing Style

Pervasive Computing Style

Technology Innovation

Business Innovation

Centralized Cloud

Centralized and Distributed Cloud

"Private" Cloud

Intentional Multicloud

Unintentional Multicloud

Fusion

Shared Services

Teams

EDGE Computing

Edge computing is a networking philosophy focused on bringing computing as close to the source of data as possible in order to reduce latency and bandwidth use.

In simpler terms, edge computing means running fewer processes in the cloud and moving those processes to local places, such as on a user's computer, an IoT device, or an edge server.

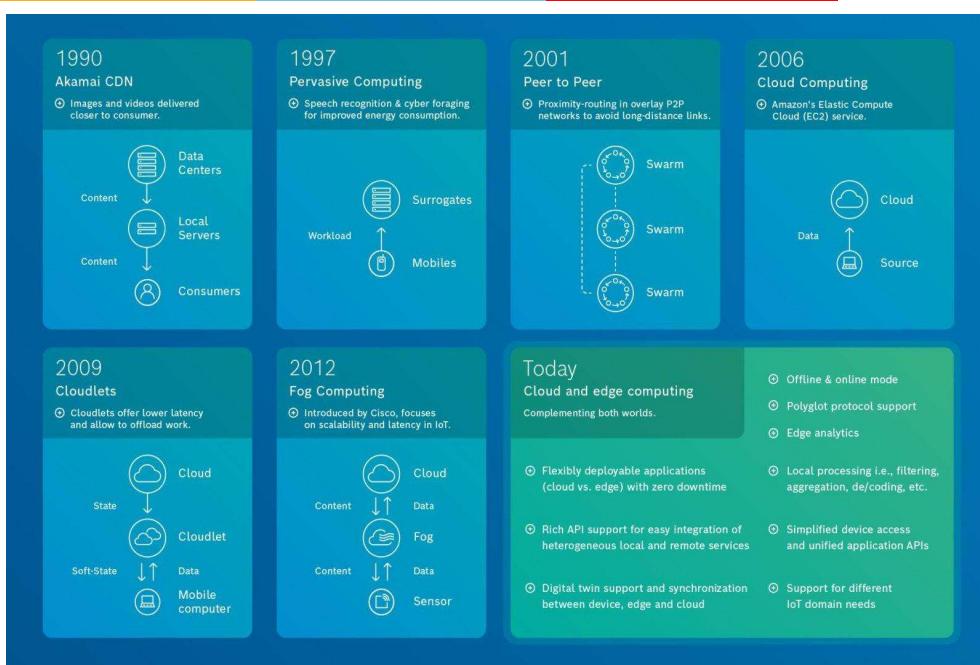
Bringing computation to the network's edge minimizes the amount of long-distance communication that has to happen between a client and server.

What is the Edge?

For Internet devices, the network edge is where the device, or the local network containing the device, communicates with the Internet. The edge is a bit of a fuzzy term; for example a user's computer or the processor inside of an IoT camera can be considered the network edge, but the user's router, ISP, or local edge server are also considered the edge. The important takeaway is that the edge of the network is geographically close to the device, unlike origin servers and cloud servers, which can be very far from the devices they communicate with.

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

EDGE Computing



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



EDGE Computing Use Cases

Edge computing can be incorporated into a wide variety of applications, products, and services. A few possibilities include:

- Security system monitoring: As described above.
- IoT devices: Smart devices that connect to the Internet can benefit from running code on the device itself, rather than in the cloud, for more efficient user interactions.
- Self-driving cars: Autonomous vehicles need to react in real time, without waiting for instructions from a server.
- More efficient caching: By running code on a CDN edge network, an application can customize how content is cached to more efficiently serve content to users.
- Medical monitoring devices: It is crucial for medical devices to respond in real time without waiting to hear from a cloud server.
- Video conferencing: Interactive live video takes quite a bit of bandwidth, so moving backend processes closer to the source of the video can decrease lag and latency.

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

Points to Ponder – ChatGPT's Prediction

1. Multi-cloud: The use of multiple cloud providers is becoming increasingly common, as companies seek to avoid vendor lock-in and take advantage of the unique capabilities of different cloud platforms.

2. Serverless Computing: Serverless computing is gaining in popularity, as it allows developers to write and run applications without having to worry about managing infrastructure.

3. Edge Computing: With the growth of IoT devices and real-time applications, there is a need for computing resources to be located closer to the source of data. Edge computing involves processing data at or near the edge of the network, rather than sending it back to the cloud.

4. Artificial Intelligence and Machine Learning: Cloud providers are incorporating AI and machine learning capabilities into their platforms, making it easier for developers to build intelligent applications.

Points to Ponder – ChatGPT's Prediction

5. Quantum Computing: Quantum computing is still in its early stages, but it has the potential to transform cloud computing by providing a significant boost in computing power.

6. Green Computing: With the growing concern for the environment, cloud providers are making efforts to reduce their carbon footprint by using renewable energy and implementing energy-efficient technologies.

7. Hybrid Cloud: As more companies adopt cloud computing, hybrid cloud solutions that combine on-premises and cloud-based infrastructure are becoming increasingly popular.

8. Security and Compliance: With the increasing number of data breaches and cyber threats, cloud providers are investing in advanced security and compliance measures to protect their customers' data.

A	
Amazon Glacier	99
Amdahls Law	219
Analysis of Security Need	186
Anatomy of a Cloud Ecosystem	202
Anatomy of a Cloud Ecosystem	233
Anatomy of Cloud	121
Application of Virtualization	69
Architecture Overview	306
Attacker Capability: Malicious Insiders	259
Attacker Capability: Outside attacker	259
AWS & Customer	80
AWS Availability Zones	78
AWS Case Study	100
AWS EBS	96
AWS EBS Types	96
AWS EFS	98
AWS EFS vs AWS EBS vs AWS S3	98
AWS Elastic Block Storage EBS	95
AWS Elastic Compute Cloud EC2	82
AWS Elastic File Storage EFS	97
AWS for the Edge	80
AWS Glacier	99
AWS Global Infrastructure	77
AWS Local Zones	78
AWS Outposts	79
AWS Reference Model	82
AWS Reference Model	103
AWS Reference Model	118
AWS Regions	77
AWS Regions & Availability Zones	102
AWS Regions & Availability Zones	118
AWS S3	94
AWS Simple Storage Service S3	93
AWS Storage	92
AWS Understanding Service Offering	76
AWS VPC	91
AWS VPC - Components	91
AWS VPC - Functioning	92
AWS Wavelength Zones	79

Azure Components	172
Azure Components?	168
Azure fabric Controller	171
Azure Introduction	170
Azure key Terms?	169
Azure PaaS Design Principles	170
Azure Runtime Environment	171
Azure Services?	168
B	
BaaS Architecture	312
BaaS BaaS Examples	313
BaaS Defined	312
BaaS Pro & Con	314
BaaS vs IaaS	313
BaaS vs Serverless	315
Benefits	240
Benefits of OpenNebula	214
Benefits of Virtualization	43
Big data	296
Binary Translation / Full Virtualization	56
Binary Translation / Full Virtualization - Cons	57
Binary Translation / Full Virtualization - Pros	56
Broad Network Access	25
Building Blocks - PaaS	162
C	
Capacity Management to meet SLA Commitments	230
Challenges for the attacker	260
Challenges to Virtualization	52
Characteristics of MT Architecture	242
Characteristics- PaaS	163
Classes of problems mapreducible	307
Cloud NIST & the AAS es	9
Cloud Origins & Motivation	6
Cloud Adoption : Reluctance	253
Cloud Advantages	27
Cloud Challenges	28
CLOUD COMPUTING	1
Cloud Computing: Security Analysis?	252
Cloud Deployment Models	15
Cloud Deployments	311

Cloud ecosystem	29
Cloud Essential Characteristics	23
Cloud Failures	30
Cloud Governance (L to R)	187
Cloud Security Issues	188
Cloud Security Issues	254
Cloud Service Models (AAS es)	10
Cloud Skills	2
Cloud vs Hosted	27
Cloud Workload An Anatomy	219
Co-Hosting Application	276
Co-Hosting Application: Issues	276
Co-Hosting Application: Solution	277
Cold Migration	135
Community Cloud	21
Companies are afraid to use clouds	253
Comparing AWS & OpenStack	113
Comparing LXC & LXD	142
Comparison OS vs Application Containers	144
Comparison with Similar Technologies	215
Components of Distributed File System	289
Connecting to EC2 Instance	87
Container Orchestration	154
Container Orchestration	157
Container Orchestration - Tools	158
Container Types Application Containers	143
Container Types OS Containers	142
Control Groups	140
Creating an EC2 Instance - Video	88
D	
Data Evolution	296
Data Security and Storage	192
Data Security and Storage	261
Demo HEAT Provisioning	128
Deployment Models in Cloud	16
Device Virtualization	65
Did You Know ?	249
Difference between VM & Container	144
Difference between VM & Container	161
Disadvantages	241
Distributed File System Challenges	289

Distributed File System- Introduction - What is Distributed File System	288
Docker Life Cycle	153
Docker Objects	149
Docker Objects - Containers	151
Docker Objects - DockerFiles	152
Docker Objects - Images	149
Docker Summary	153
Dockers	145
Dockers Components Client	148
Dockers Components Registries	148
Dockers Components - Daemon	147
Dockers - Architecture	147
Dockers - Introduction	146
Dockers - Motivation	145
Dockers - Motivation	160
E	
EC2 Accessing over Web	88
EC2 Lifecycle	89
EC2 Placement Groups	90
EC2 Tenancy Options	89
EC2 - Amazon Machine Image (AMI)	85
EC2 - AMI Types	85
EC2 - Instance Type	84
EC2 - Instance Type & AMI	83
EC2 Introduction	83
EDGE Computing	323
EDGE Computing Use Cases	324
Emulation	55
Enter Distributed Management of Virtual Resources	210
Essential Characteristics	24
F	
FaaS	319
FaaS Benefits	320
Facts about Cloud	5
Facts about Cloud - Now	5
Familiar Use Cases	4
Fault Tolerance in MapReduce	308
Features of OpenNebula	214
File System- Introduction	288
Files on GFS	293

Function as a Service	318
FUTURE DIRECTIONS	136
Future Directions	322
G	
Generic 3-tier Web Application	116
GFS - Chunk Replica	295
GFS - Chunks	294
GFS - Master	293
GFS - Metadata	294
GFS - Operational Logs	295
GFS Architecture	292
GFS Assumptions	291
GFS Design Overview	292
GFS is not suited	291
Google File System	290
H	
H/W Assisted Virtualization	58
Hadoop	297
Hadoop – HDFS Design	298
Hadoop Distributed File System	303
HDFS BLOCK CACHING	302
HDFS BLOCKS	300
HDFS FEDERATION	303
HDFS File Read	304
HDFS MR	305
HDFS Name nodes and data nodes	301
HDFS REDUNDANCY (High Availability)	302
HFDS YARN	304
History of Open stack	104
How do we define Cloud ?	8
How Haizea Scheduler Works	226
How To ? Find the Right Balance	205
Hybrid Cloud	19
Hypervisor	36
Hypervisor - Samples	37
Hypervisor Architecture	38
Hypervisor Goals	36
Hypervisor Techniques	42
Hypervisor Types	38

I		
IaaS Key Terms		74
IAAS MT Model		239
IaaS - Motivations		74
IAC		155
Image Management in OpenNebula		213
Impact of cloud on the governance structure of IT organizations		252
Imperative vs Declarative Model in IAC		156
Importance of Capacity Management		204
Importance of Capacity Management		234
Infrastructure as a Service		10
Infrastructure as a Service		73
Infrastructure SLA		279
Infrastructure SLA vs Capacity Management		279
Infrastructure SLA's		230
Introducing Amazon Web Service		75
Introducing Network Functions Virtualization - NFV		109
Introducing Platform as a Service		162
Introducing Software as a Service		174
K		
Kubernetes Introduction Video		158
L		
Lack of Trust in the Cloud		255
Leasing Schedule Advanced Reservation (AR)		229
Leasing Schedule Best Effort Lease (BEL)		228
Linux Containers		139
Linux Containers LXC		139
Linux Containers LXD		141
Live Migration Effect on a Running Web Server		134
Live Migration Process		132
Live Migration Technique		133
Live Migration Vendor Implementations Example		134
Live Migration Xen Hypervisor		130
Loss of Control Trust Issues		189
Loss of Control in the Cloud		254
M		
MapReduce (Data Processing Framework)		305
MapReduce Processing flow		306
MBaaS		321

MBaaS Architecture	321
Memory Virtualization	64
Metered by Use	26
Microservice	319
Minimalistic OS	143
Minimize Lack of Trust: Certification	196
Minimize Lack of Trust: Certification	263
Minimize Lack of Trust: Policy Language	196
Minimize Lack of Trust: Policy Language	263
Minimize Loss of Control: Monitoring	197
Minimize Loss of Control: Monitoring	264
Minimize Loss of Control: Access Control	198
Minimize Loss of Control: Access Control	265
Minimize Loss of Control: Monitoring (Cont.)	264
Minimize Loss of Control: Utilize Different Clouds	265
Minimize Multi-tenancy	199
Minimize Multi-tenancy	266
Mission Statement	105
Mobile BaaS	320
Module 1 - Introduction to Cloud Computing	2
Module 2 - Virtualization Techniques and Types	32
Module 3 - Infrastructure as a Service	72
Module 4 - Containers	137
Module 5 - Platform as a Service and Software as a Service (PaaS & SaaS)	161
Module 6 Capacity Management and Scheduling in cloud computing	202
Module 7 - Issues and Challenges: Availability, Multi-Tenancy, Security and SLA	236
Module 8 - DFS, GFS, MR and Hadoop	287
Motivation towards Containers	138
Motivation towards Containers	159
Motivations & Origins	33
MT Issues in Cloud	189
MT- Different Levels	242
MT- Different Levels	269
Multi Tenancy	236
Multi Tenancy: Resource Sharing	245
Multi Tenancy: Resource Sharing	248
Multi Tenant vs Multi Instance	238
Multi Tenant vs Multi Instance	269
Multi-tenancy Issues in the Cloud	255
N	
Netflix	100

Network Virtualization	63
Networking in AWS - VPC	90
Networking OpenNebula	213
NFV Origins & Motivations	110
NFV Architecture	112
NFV Benefits	113
NFV Comparison	112
NIST 3-4-5 Rule for Cloud	9
NIST Definitions	31
O	
Objective	186
On Demand Self Service	25
OpenNebula VIM	211
OpenStack	103
OpenStack Case Study	115
OpenStack Projects	106
OpenStack Projects	107
OpenStack Reference Model	107
OpenStack Reference Model	119
OpenStack Services	108
OpenStack Services Used	116
Origins	6
Overview	114
P	
PaaS AWS Examples	166
PaaS Best Practices	165
PAAS MT Model	239
PaaS Vendors (Popular)	172
PaaS Advantages	164
PaaS Disadvantages	164
PaaS vs IaaS	163
Para Virtualization	57
Paradigms	7
PART II: SECURITY AND PRIVACY ISSUES IN CLOUD COMPUTING - BIG PICTURE	260
PART III. POSSIBLE SOLUTIONS	262
Perception	311
Platform as a Service	12
Platform as a Service - Summary	166
Points to Note	68
Private Cloud	18

Pros & Cons of IaaS	75
Provisioning to meet SLA	223
Pubic Cloud	16
Q	
Quick Comparison	23
R	
Rapid Elasticity	26
Reservation Based Provisioning	223
Resource pooling	24
Resource Sharing Approaches	245
Resource Sharing in VM - CPU	40
Resource Sharing in VM - IO	41
Resource Sharing in VM - Memory	41
S	
SAAS MT Model	240
SaaS Adantages	178
SaaS Applicability Scenarios 1 2 3	183
SaaS Architecture	175
SaaS Delivery Model	175
SaaS Model Comparison	176
SaaS Motivations	174
SaaS Providers	178
SaaS Providers at a Glance	179
SaaS User Benefits	180
SaaS Vendor Benefits	181
SaaS vs PaaS vs IaaS APPLICABILITY	184
SaaS vs PaaS vs IaaS COMPARISION	184
SaaS vs PaaS vs IaaS COMPARISION	201
Sample MT Architecture	248
Scheduling Techniques	220
Scheduling Techniques Existing Approaches	220
Scheduling Techniques VM Overheads	222
Scheduling Techniques for Advance Reservation of Capacity	224
Scheduling VM Workloads	215
SEAY-ZG527 CLOUD COMPUTING	250
SECURITY	251
Security in MT	244
Security in the Cloud	185
Security in the Cloud Possible Solutions	193

Security Issues in the Cloud	262
Security Issues in the Cloud My Diagnosis CLOUD SECURITY ISSUES DETECTION DOES NOT MINIMIZE THE RISK, YOU NEED TO REMEDY.	194
Security Issues in the Cloud Remedy	194
Security Issues in the Cloud The What	193
Server Virtualization	61
Server Virtualization - Benefits	62
Serverless Advantages	317
Serverless Computing	314
Serverless Disadvantages	318
Serverless on AWS	316
Session Agenda	1
Session Agenda	310
SKI Virtualization	59
SLA	274
SLA Types	278
SLA Cloud Application LC Management	283
SLA LC Management Cloud Applications	282
SLA Life Cycle	280
SLA Management	270
SLA Role in High Availability	271
SLA vs SLO	273
So What is the Solution	210
So What is the way out? VIM	207
So Whats the Way? Find the Right Balance	204
Software as a Service	13
Software as a Service	173
Software as a Service	182
Software as a Service - Summary	200
Solution Haizea Scheduler	225
Some Facts	237
Some Facts	268
Some Facts about MR	309
Stages of VM Life Cycle in OpenNebula	212
Steps for HA	272
Storage Migration	135
Storage Types	93
Storage Virtualization	62
Storage Virtualization - Benefits	63
Support for Customization	246

T		
Taxonomy of Fear		190
Taxonomy of Fear - CIA		256
Taxonomy of Fear (cont.)		256
Technology Trends		69
Terms to Remember		8
The 3 Initialisms		273
The 3 Initialisms		274
The Right SaaS Model?		183
Threat Model		191
Threat Model		257
Threat Sources		191
Traditional SLO Management Approaches		277
True SaaS Applicability		201
Type 1 & 2 Hypervisor At a Glance		50
Type 1 Hypervisor		47
Type 1 Hypervisor Cons		48
Type 1 Hypervisor Pros		48
Type 2 Hypervisor		49
Type 2 Hypervisor - CONs		50
Type 2 Hypervisor - PROs		49
U		
Understanding Hypervisor		46
Understanding Vagrant		127
Using AWS - Connecting		81
Using AWS - Video		81
V		
Video BOCHS		37
Video Virtualization		33
Virtual Infrastructure Management		206
Virtual Infrastructure Manager (VIM)		208
Virtualization		61
Virtualization		71
Virtualization		137
Virtualization Advantages		65
Virtualization Advantages		66
Virtualization Approaches		54
Virtualization Architecture		35
Virtualization Comparison		60

Virtualization Comparison	71
Virtualization Evolution	51
Virtualization Evolution	53
Virtualization History	32
Virtualization Summary	44
Virtualization Summary	66
Virtualization Types	60
Visualization for a New Paradigm	73
VM Lifecycle	125
VM Management	121
VM Management in OpenNebula	212
VM Migration	128
VM Provisioning process	126
VM Provisioning using templates	127
VMware's Solution	53
W	
What are Containers?	138
What are Containers?	160
What are the Challenges?	203
What are the Challenges?	234
What are we Talking about	217
What do we manage in Private Cloud?	123
What do we manage in Public Cloud?	122
What is Haizea Scheduler	225
What is a Cloud Workload	216
What is a Cloud Workload	235
What is a Virtual Infrastructure Manager?	208
What is a Virtual Infrastructure Manager?	235
What is AWS	76
What is AWS	102
What is AWS	117
What is Big data?	297
What is Cloud Computing Your Opinion ?	3
What is Hypervisor	46
What is IaaS?	72
What is Microsoft Azure?	167
What is Multi Tenancy?	236
What is Multi Tenancy?	267
What is NFV	111
What is NFV?	109
What is OpenNebula	211

What is OpenStack?	104
What is OpenStack?	119
What is Privacy?	192
What is Privacy?	261
What is Resource Management	124
What is SLA or Service Level Agreement	271
What is the issue?	258
What is Virtual Infrastructure Management	206
What is Virtualization?	34
What is Virtualization?	45
What is Virtualization?	70
What is VM Live Migration	130
What is VM Migration	129
What Next in Cloud	322
What the Future Holds?	188
What you get with OS	105
Whats in the Cloud ??	4
Where do we Run Workloads?	217
Which AAS?	14
Who Can Use PaaS	165
Who Manages the AAS es?	15
Why a Virtual Infrastructure Manager?	209
Why Capacity Management in Cloud?	203
Why is it Called Cloud	3
Why Reluctance ?	187
Why Resource Management	123
Why use Hypervisor	47
Windows Azure	167
Workload Challenges	218
X	
x86 Architecture	51