

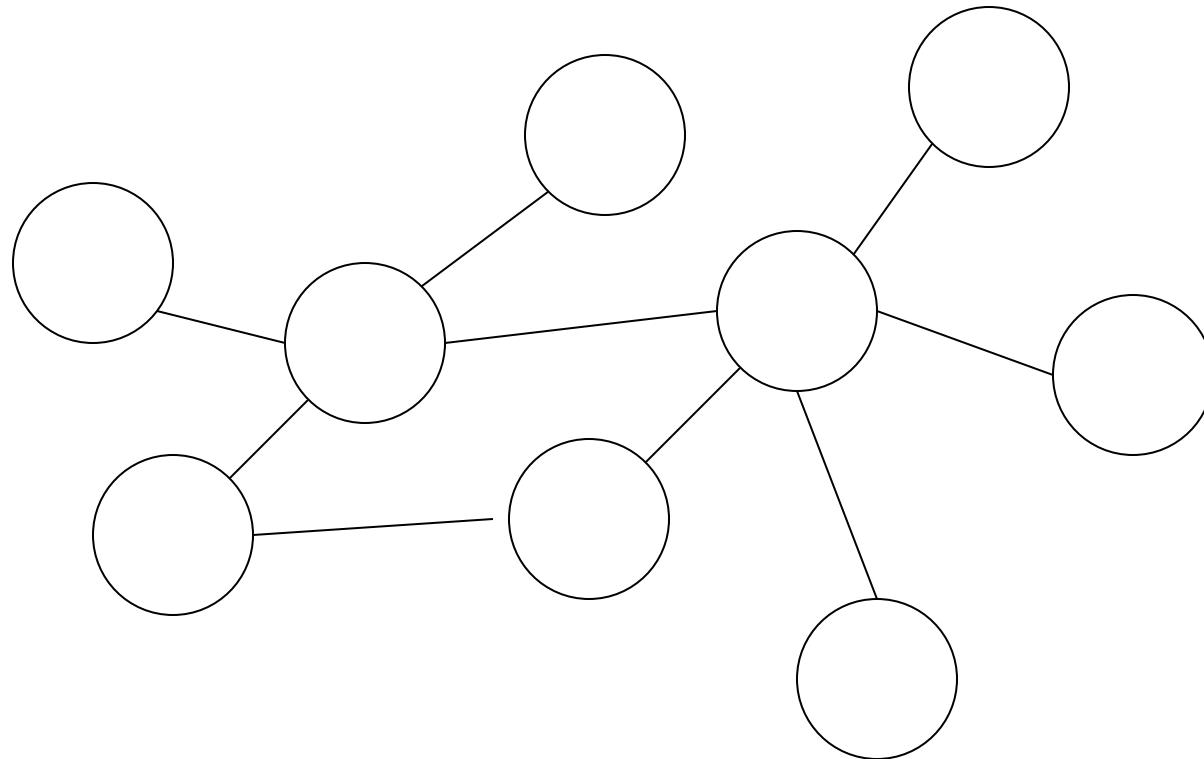
Unit 1

Networking Fundamentals

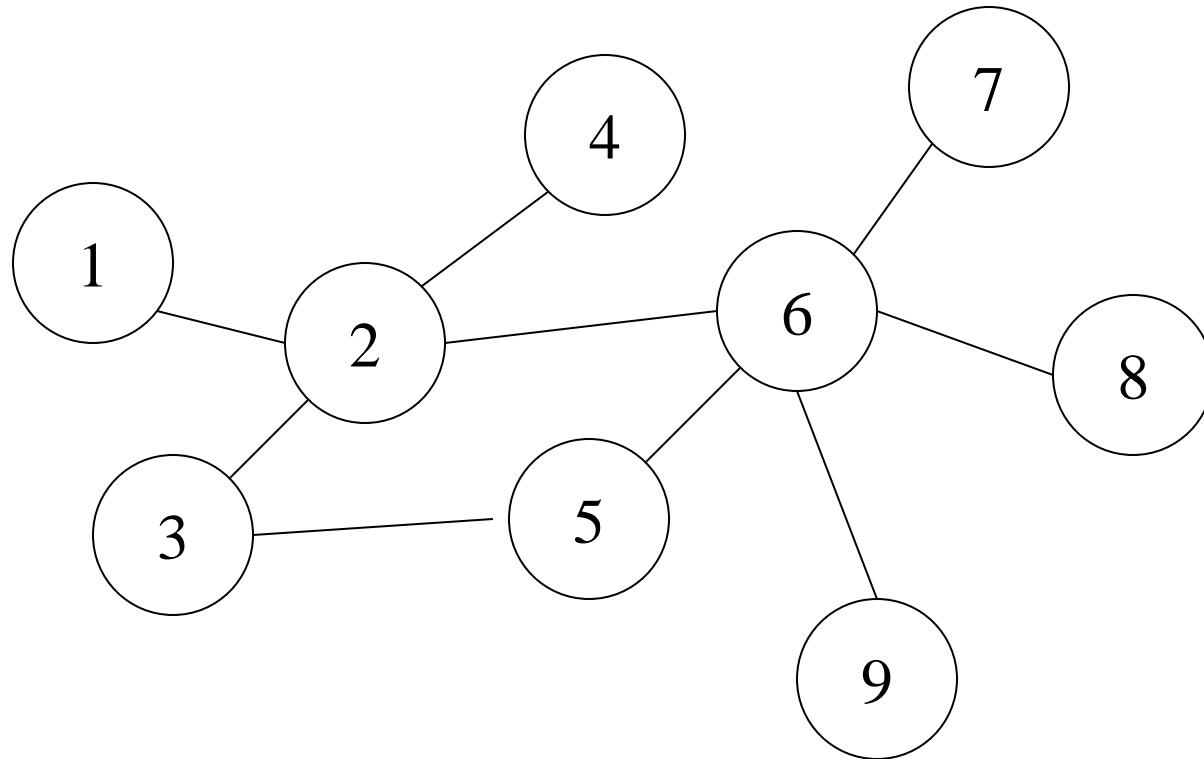
Basics

- Network – collection of nodes and links that cooperate for communication
- Nodes – computer systems
 - Internal (routers, bridges, switches)
 - Terminal (workstations)
- Links – connections for transmitting data
- Protocol – standards for formatting and interpreting data and control information

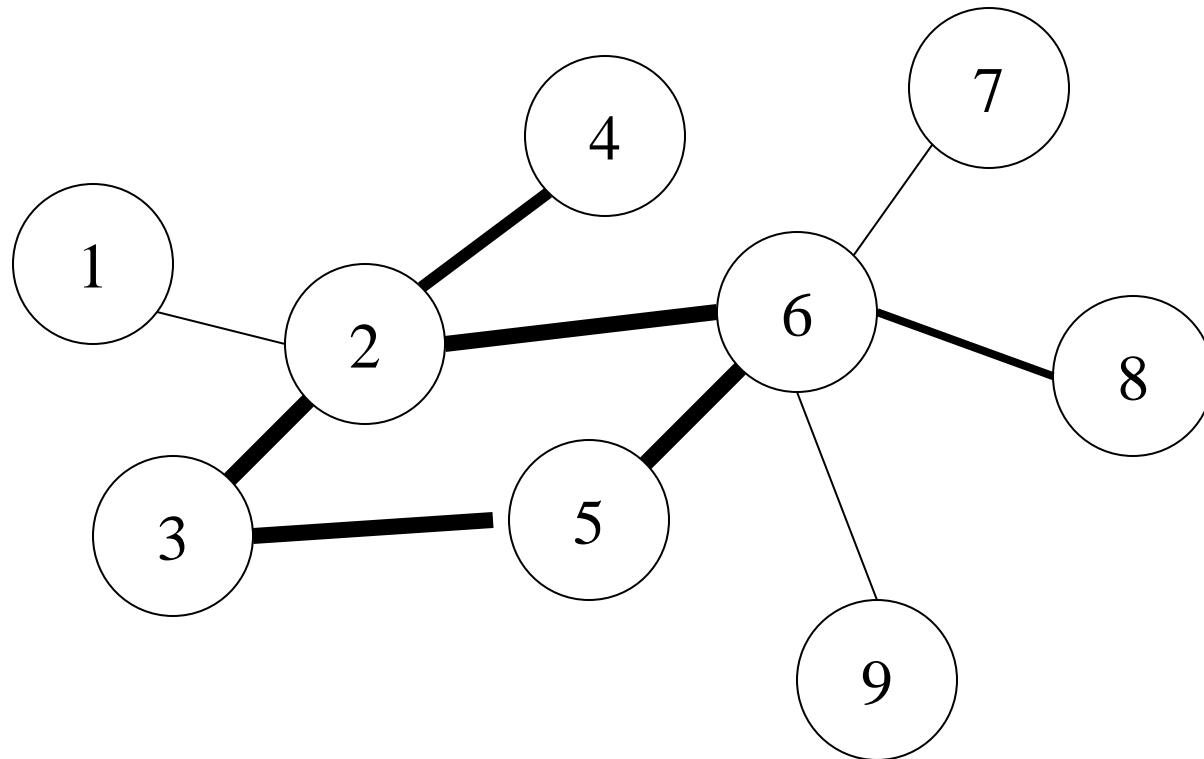
Network = {nodes and links}



Nodes have addresses



Links have bandwidths and latencies



Wires aren't perfect

- Attenuation (resistance)
 - degrades quality of signal
- Delay (speed of light * 2/3)
 - speed of light:
 - 8ms RTT coast-to-coast
 - 8 minutes to the sun
- Noise (microwaves and such)
- Nodes aren't perfect either
 - Unreliability is pervasive!

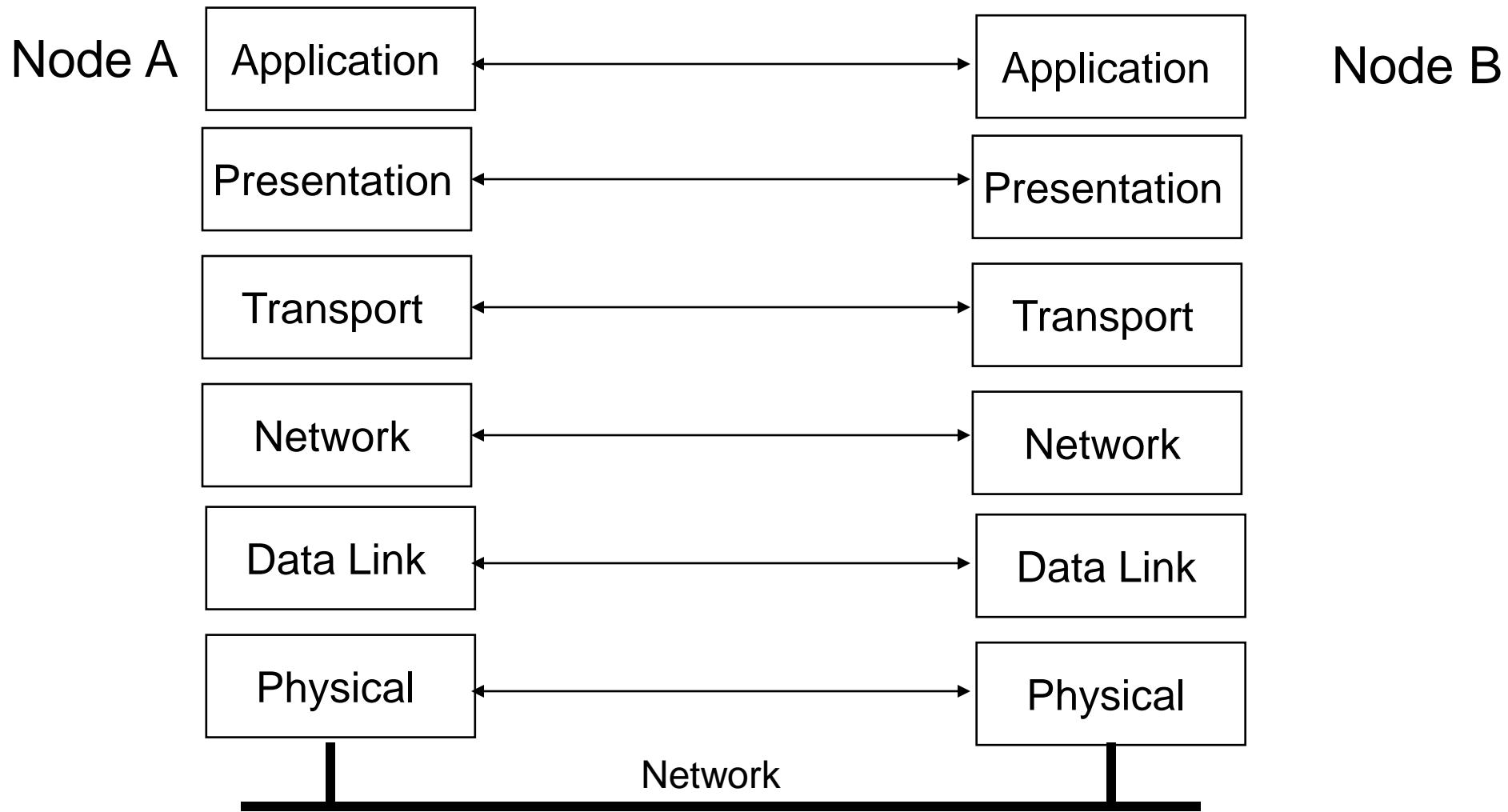
Getting Data Across (imperfect wires)

- Split up big files into small pieces
 - the pieces are called **packets**
- Each packet (~ 1500 bytes) is sent separately
 - packets can be corrupted
 - noise, bugs
 - packets can be dropped
 - corrupted, overloaded nodes
 - packets can be reordered
 - retransmission + different paths
- Allows packets from different flows to be multiplexed along the same link

Layers

- Each layer abstracts the services of various lower layers, providing a uniform interface to higher layers.
- Each layer needs to know:
 - How to interpret a packet's payload
 - e.g., protocol numbers
 - How to use the services of a lower layer

OSI Levels



Layers

OSI Reference

Application
Presentation

Session

Transport

Network

Data-Link

Physical

Reality

HTTP

TCP

IP

Ethernet

Twisted Pair

Packet Format



The Internet Protocol (IP)

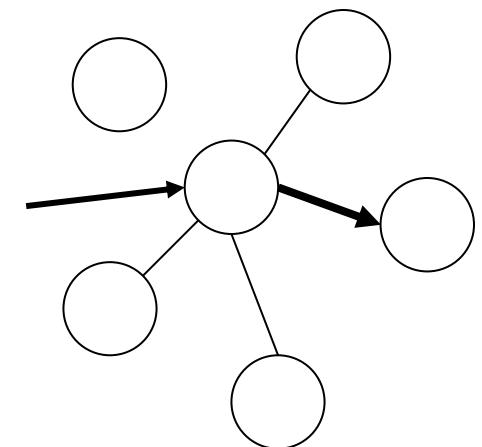
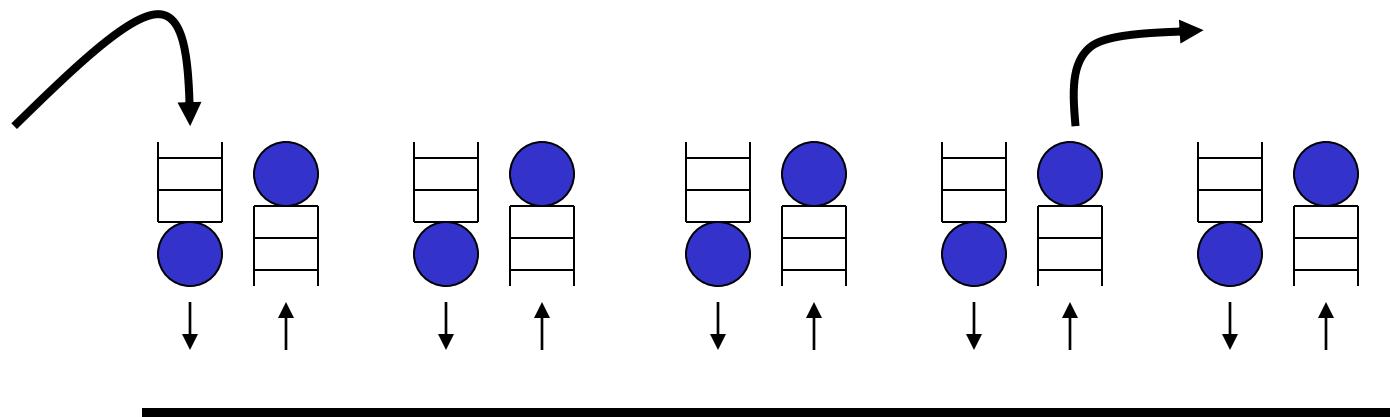
- Connects disparate networks
 - Single (hierarchical) address space
 - Single network header
- Assumes data link is unreliable
- Provides unreliable service
 - Loss: A B D E
 - Duplication: A B B C D E
 - Corruption: A Q C D E
 - Reordering: A C D B E

IP Addresses

- 32 bits long, split into 4 octets:
 - For example, 128.95.2.24
- Hierarchical:
 - First bits describe which network
 - Last bits describe which host on the network
- UW subnets include:
 - 128.95, 140.142...
- UW CSE subnets include:
 - 128.95.2, 128.95.4, 128.95.219...

Packet Forwarding

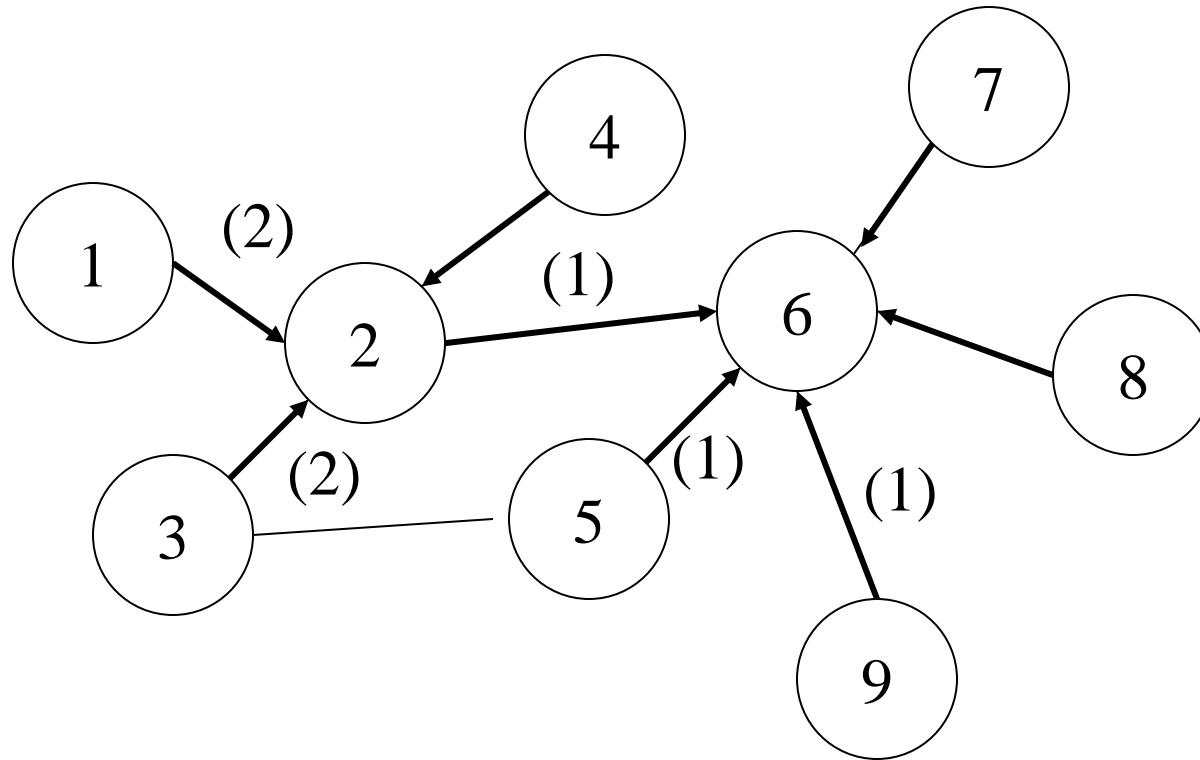
- Buffer incoming packets
- Decide which output link
- Buffer outgoing packets
- Send packet



Routing

- How do nodes determine which output link to use to reach a destination?
- Distributed algorithm for converging on shortest path tree
- Nodes exchange reachability information:
 - “I can get to 128.95.2.x in 3 hops”

Shortest path tree



(x) Is the cost to get to 6. The metric (cost per link) here is 1.
Simple algorithm: 6 broadcasts “I’m alive” to neighbors.
Neighbors send “I can get to 6 in 1 hop”, etc.

Route Aggregation

- What hierarchical addressing is good for.
- UW routers can advertise 128.95.x.x
 - instead of 128.95.2.x, 128.95.3.x, ...
- Other routers don't need forwarding table entries for each host in the network.

Routing Reality

- Routing in the Internet connects Autonomous Systems (AS's)
 - AT&T, Sprint, UUNet, BBN...
- Shortest path, sort of... money talks.
 - actually a horrible mess
 - nobody really knows what's going on
 - get high \$\$ job if you are a network engineer that messes with this stuff

TCP Service Model

- Provide reliability, ordering on the unreliable, unordered IP
- Bytestream oriented: when you send data using TCP, you think about bytes, not about packets.

TCP Ports

- Connections are identified by the tuple:
 - IP source address
 - IP destination address
 - TCP source port
 - TCP destination port
- Allows multiple connections; multiple application protocols, between the same machines
- Well known ports for some applications: (web: 80, telnet: 23, mail:25, dns: 53)

TCP's Sliding Window

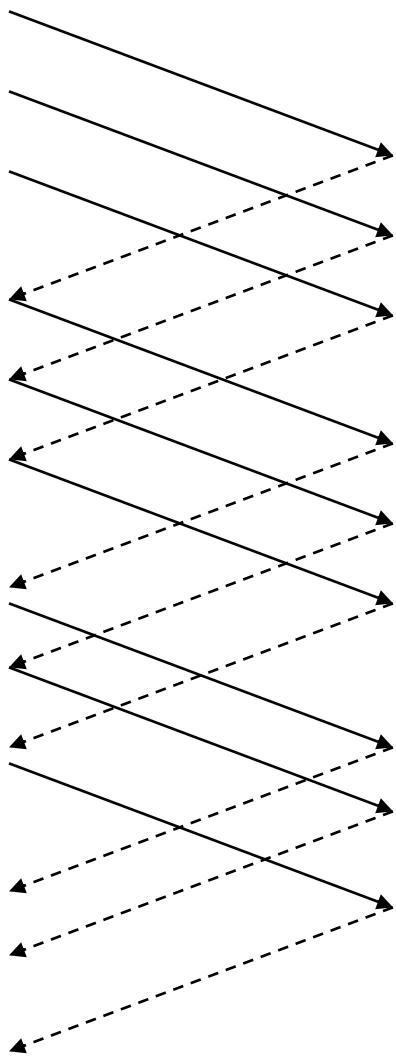
- Simple reliability:
 - Send one packet, wait for acknowledgment, then send the next...
- Better performance:
 - Keep several unacknowledged packets in the network (a window)

Sliding Window Example

Send 1
Send 2
Send 3

Send 4
Send 5
Send 6

Send 7
Send 8
Send 9



Ack 1
Ack 2
Ack 3

Ack 4
Ack 5
Ack 6

Ack 7
Ack 8

- Window size = 3
- Can send up to three packets into the network at a time.
- Each packet has a sequence number for ordering

TCP's Congestion Control

- How big should the window be?
- Performance is limited by:
 - (window size) / round trip time
 - Performance of bottleneck link (modem?)
- If window is too small, performance is wasted.
- If window is too big, may overflow network buffers, causing packet loss.

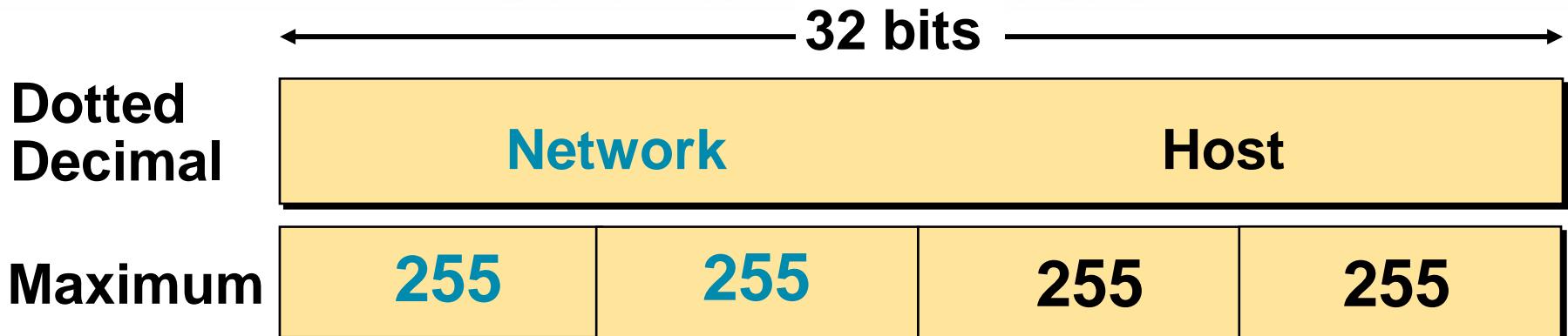
Steps for a web access

- Name lookup
 - Client to local DNS server
 - Local DNS may return a cached binding, or lookup the name for itself
- TCP Connection setup
 - Client to remote IP, port 80
- Send HTTP request
 - “GET /index.html”
- Receive HTTP response
 - “blah blah blah” maybe several packets
- TCP Connection teardown

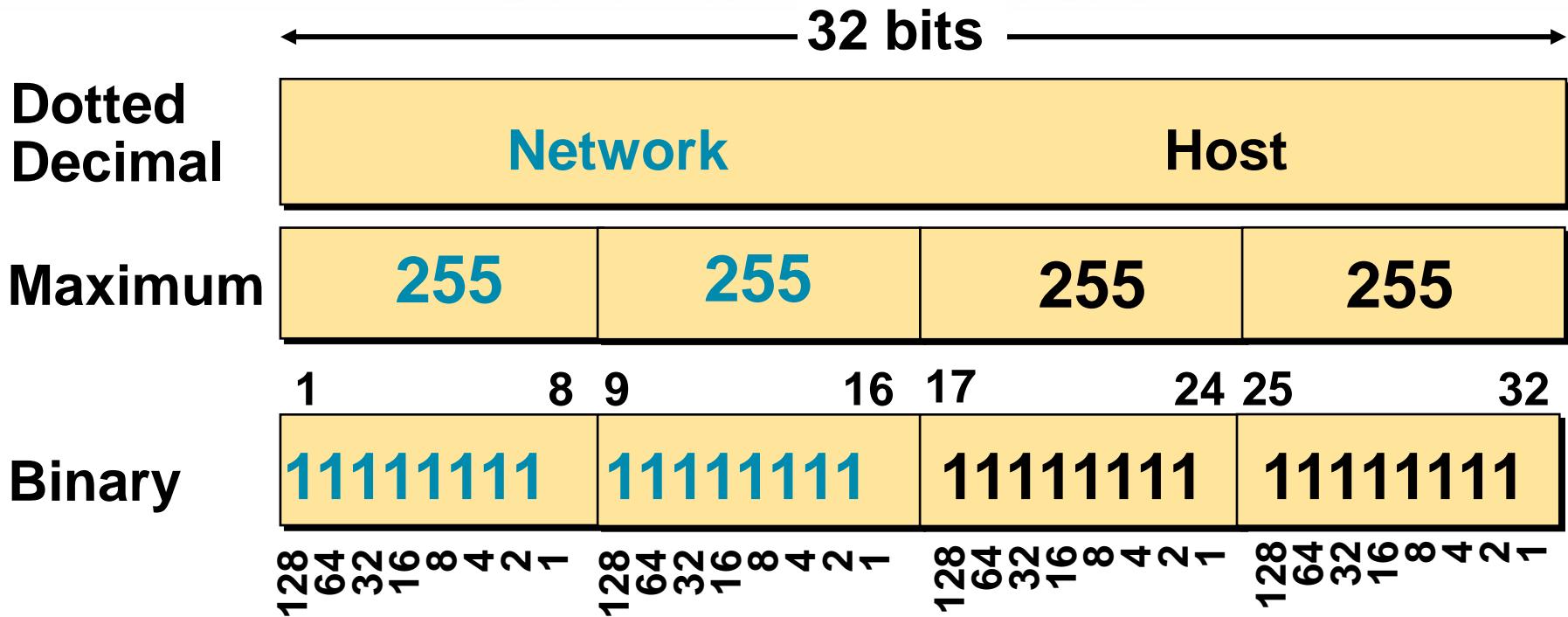
HTTP 1.1

- Incremental improvements
- “Persistent connections” allow multiple requests over the same connection
 - Web transfers are often small
 - Avoid connection setup and teardown overhead
 - TCP is better the longer you use it: it learns how fast to send to get best performance without overflowing buffers.

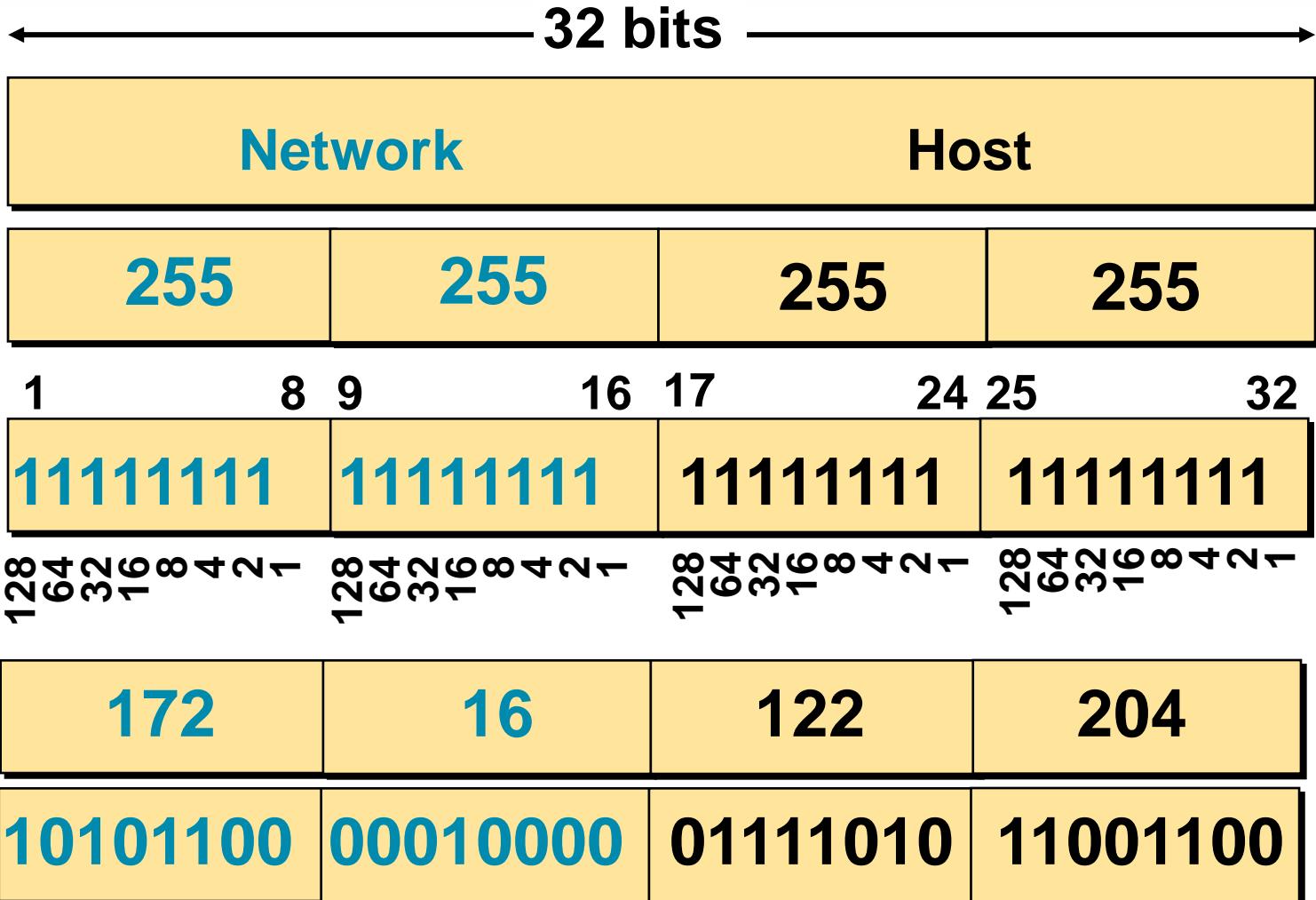
IP Addressing



IP Addressing



IP Addressing



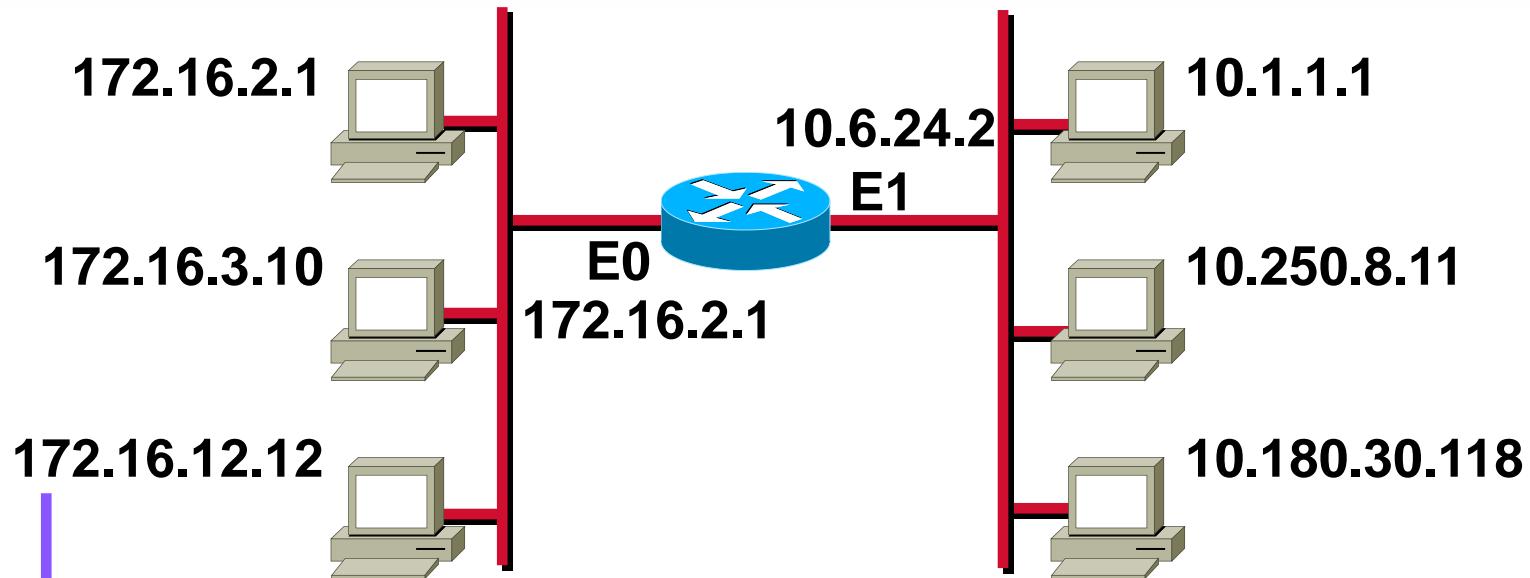
IP Address Classes

	8 bits	8 bits	8 bits	8 bits
Class A:	Network	Host	Host	Host
Class B:	Network	Network	Host	Host
Class C:	Network	Network	Network	Host
Class D:	Multicast			
Class E:	Research			

IP Address Classes

Bits:	1	8 9	16 17	24 25	32
Class A:	0NNNNNNN	Host	Host	Host	
	Range (1-126)				
Class B:	10NNNNNN	Network	Host	Host	
	Range (128-191)				
Class C:	110NNNNN	Network	Network	Host	
	Range (192-223)				
Class D:	1110MMMM	Multicast Group	Multicast Group	Multicast Group	
	Range (224-239)				

Host Addresses



172.16 . 12 . 12
Network Host

Routing Table	
Network	Interface
172.16.0.0	E0
10.0.0.0	E1

Determining Available Host Addresses

Network		Host		
172	16	0	0	
10101100	00010000	00000000	00000000	
		00000000	00000001	1
		00000000	00000011	2
				3
				⋮
	11111111	11111101	65534	
	11111111	11111110	65535	
	11111111	11111111	65536	
			- 2	
			65534	
$2^N - 2 = 2^{16} - 2 = 65534$				

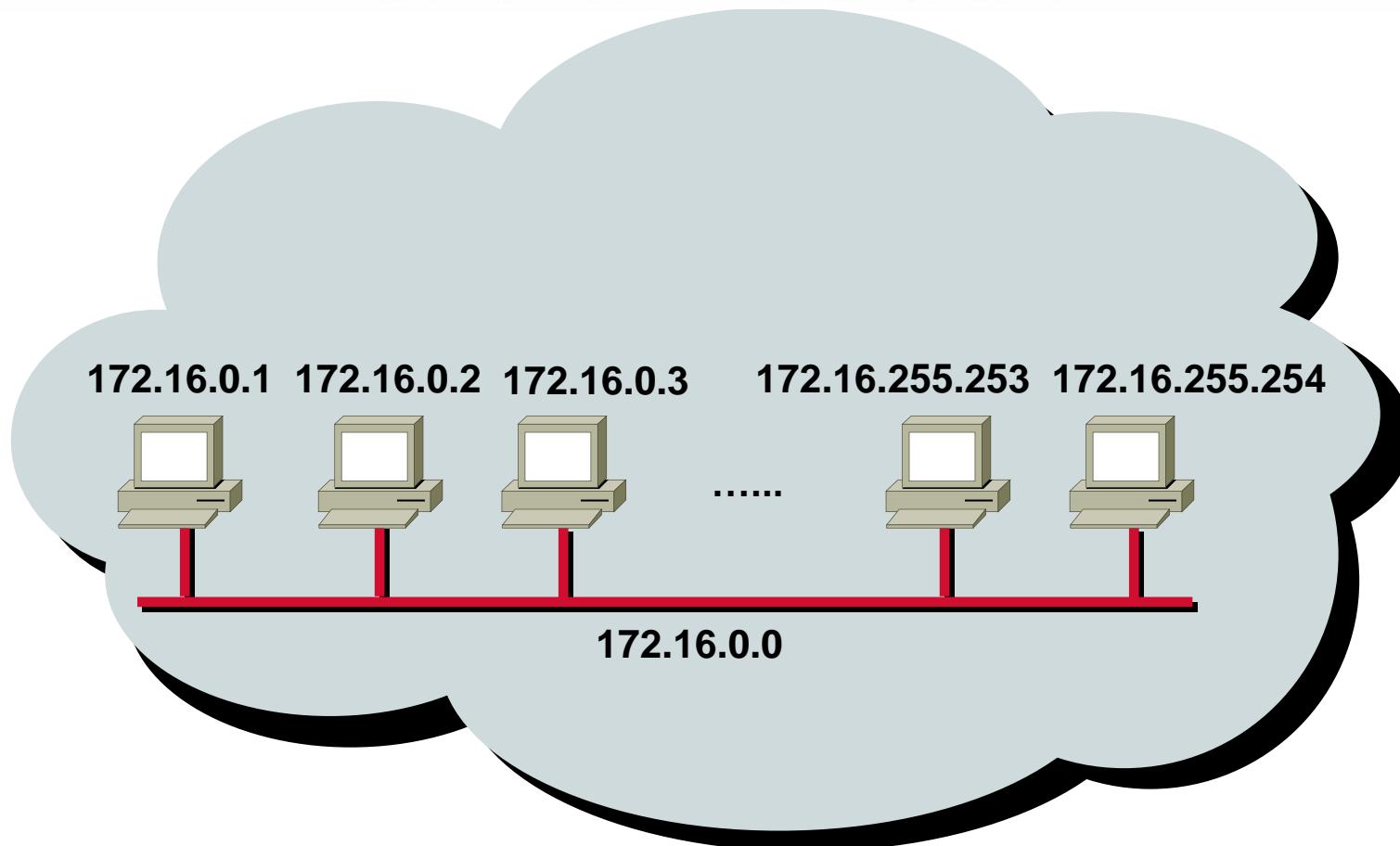
IP Address Classes Exercise

Address	Class	Network	Host
10.2.1.1			
128.63.2.100			
201.222.5.64			
192.6.141.2			
130.113.64.16			
256.241.201.10			

IP Address Classes Exercise Answers

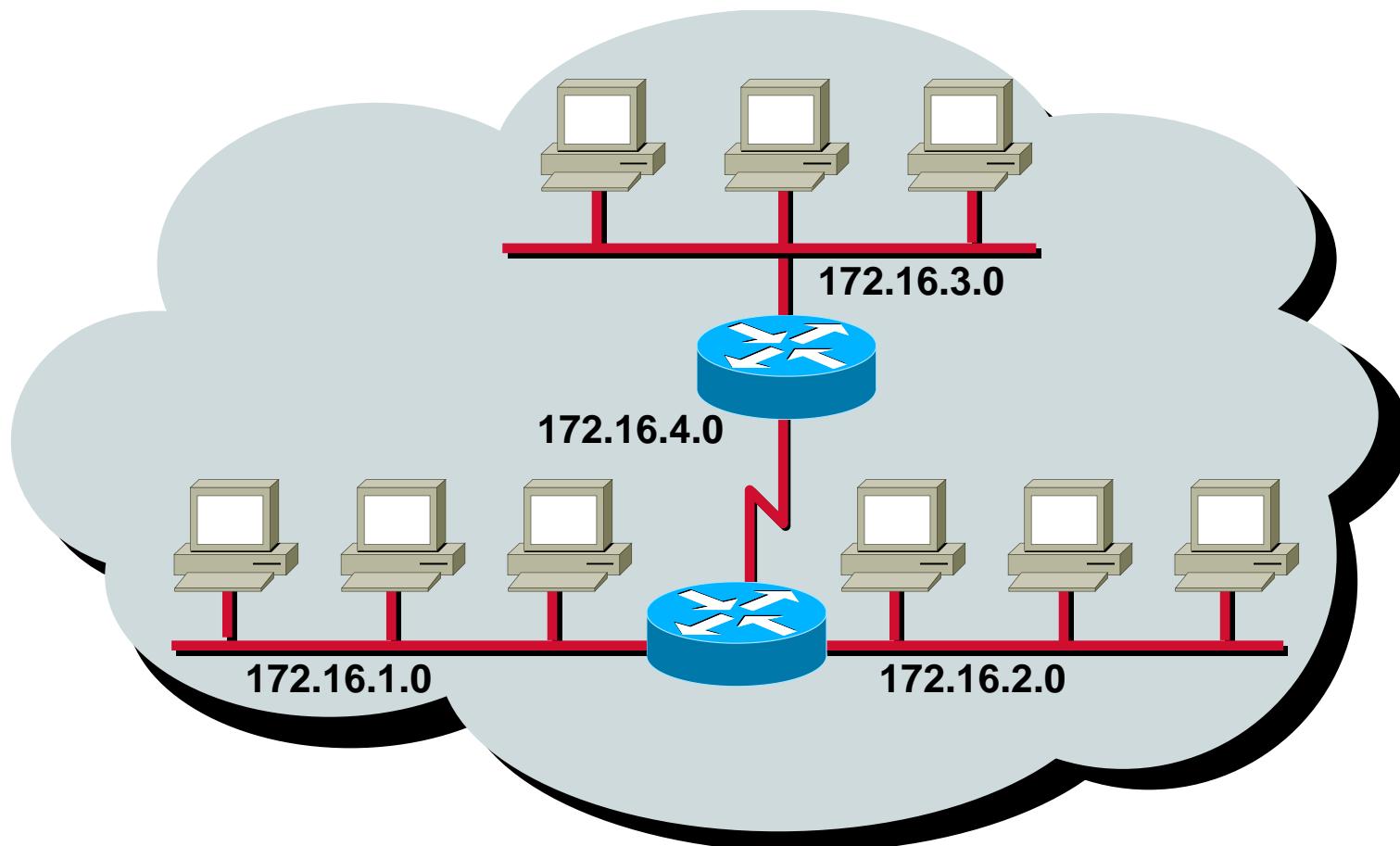
Address	Class	Network	Host
10.2.1.1	A	10.0.0.0	0.2.1.1
128.63.2.100	B	128.63.0.0	0.0.2.100
201.222.5.64	C	201.222.5.0	0.0.0.64
192.6.141.2	C	192.6.141.0	0.0.0.2
130.113.64.16	B	130.113.0.0	0.0.64.16
256.241.201.10	Nonexistent		

Addressing without Subnets



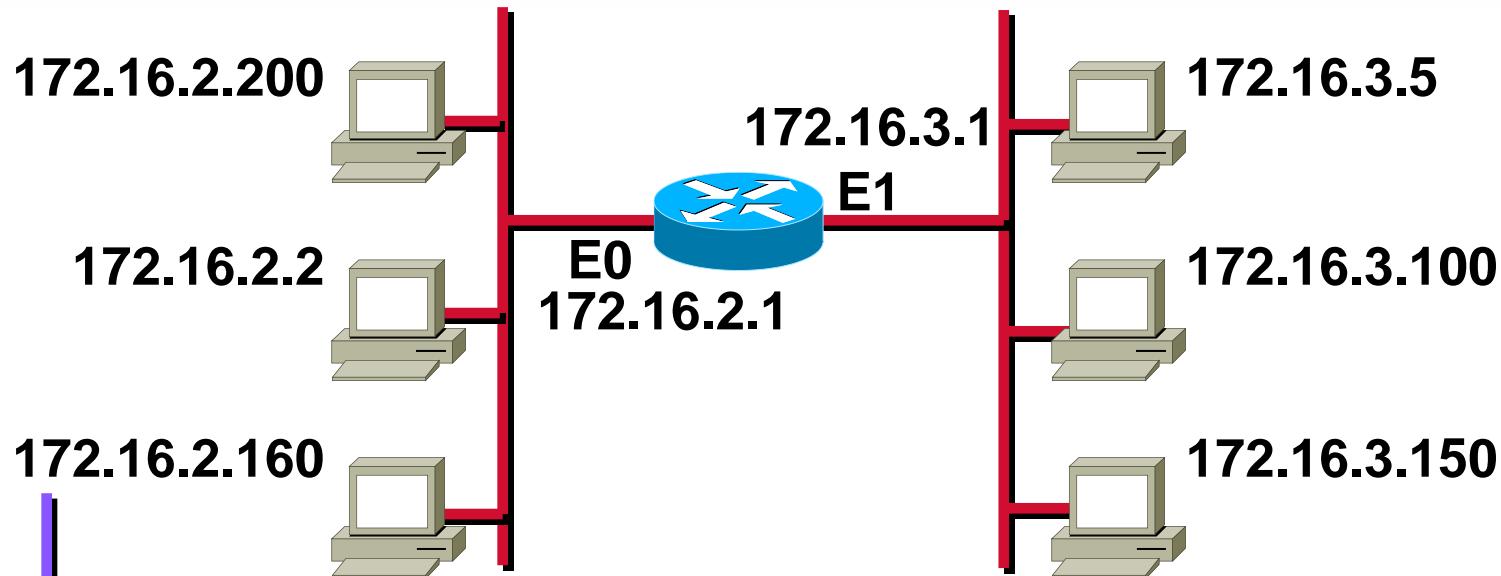
Network 172.16.0.0

Addressing with Subnets



Network 172.16.0.0

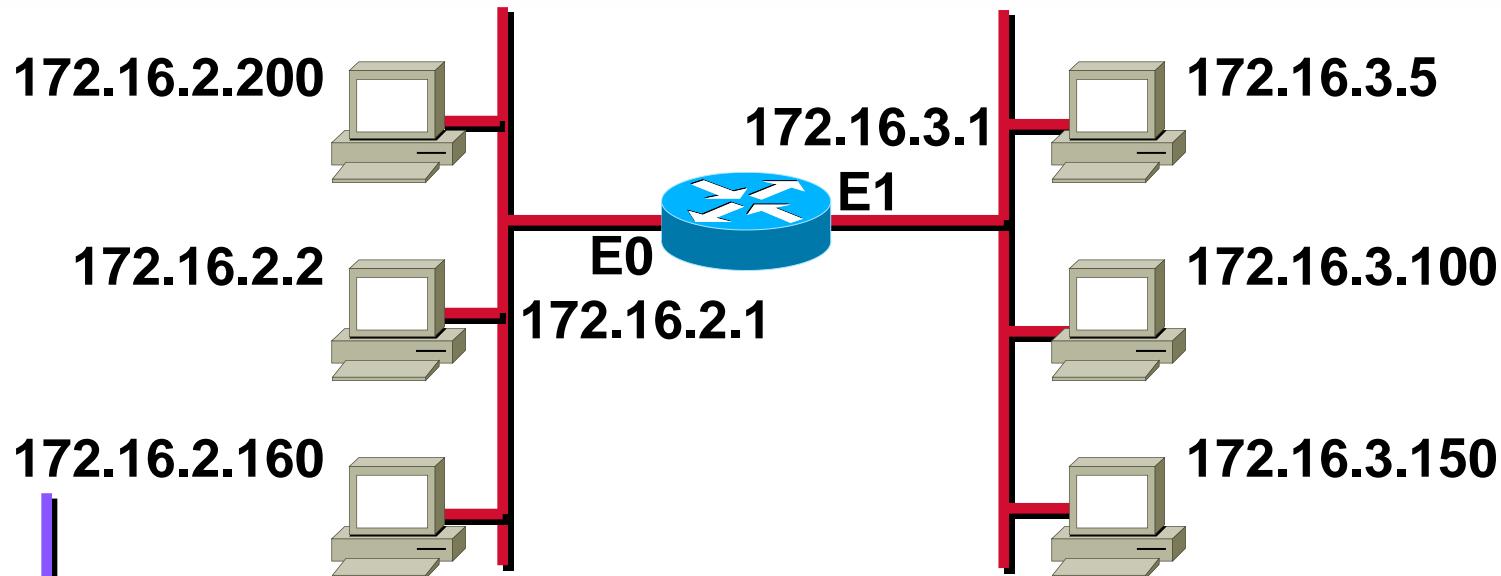
Subnet Addressing



172.16 . 2 . 160
Network Host

Network	Interface
172.16.0.0	E0
172.16.0.0	E1

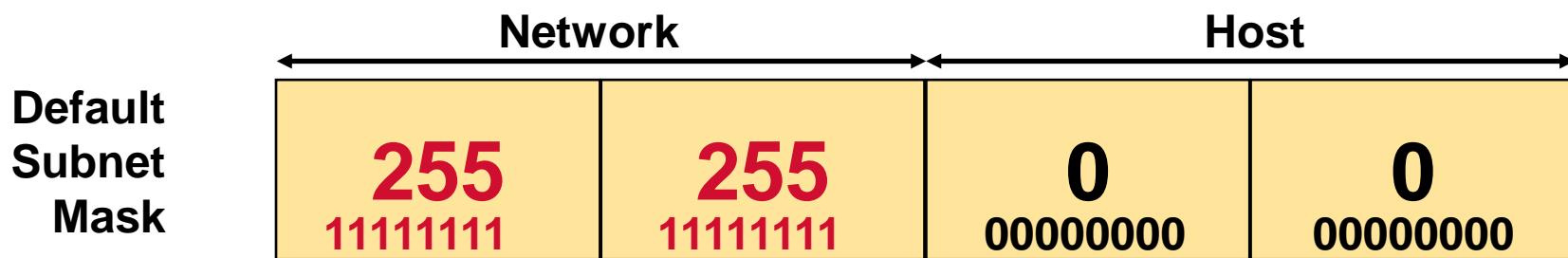
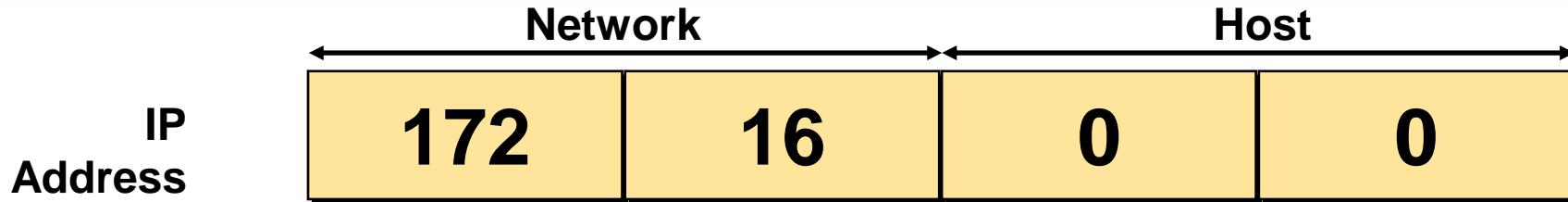
Subnet Addressing



172.16 . 2 . 160
Network Subnet Host

Network	Interface
172.16.2.0	E0
172.16.3.0	E1

Subnet Mask



Also written as “/16” where 16 represents the number of 1s in the mask.



Also written as “/24” where 24 represents the number of 1s in the mask.

Decimal Equivalents of Bit Patterns

128	64	32	16	8	4	2	1		=	
1	0	0	0	0	0	0	0		=	128
1	1	0	0	0	0	0	0		=	192
1	1	1	0	0	0	0	0		=	224
1	1	1	1	0	0	0	0		=	240
1	1	1	1	1	0	0	0		=	248
1	1	1	1	1	1	0	0		=	252
1	1	1	1	1	1	1	0		=	254
1	1	1	1	1	1	1	1		=	255

Subnet Mask without Subnets

	Network		Host	
172.16.2.160	10101100	00010000	00000010	10100000
255.255.0.0	11111111	11111111	00000000	00000000
	10101100	00010000	00000000	00000000
Network Number	172	16	0	0

Subnets not in use—the default

Subnet Mask with Subnets

	Network		Subnet	Host
172.16.2.160	10101100	00010000	00000010	10100000
255.255.255.0	11111111	11111111	11111111	00000000
	10101100	00010000	00000010	00000000

128
192
224
240
248
252
254
255

Network Number

172	16	2	0
-----	----	---	---

Network number extended by eight bits

Subnet Mask with Subnets (cont.)

	Network	Subnet	Host	
172.16.2.160	10101100	00010000	00000010	10100000
255.255.255.192	11111111	11111111	11111111	11000000
	10101100	00010000	00000010	10000000

128 192 224 240 248 252 254 255

128 192 224 240 248 252 254 255

Network Number

172	16	2	128
-----	----	---	-----

Network number extended by ten bits

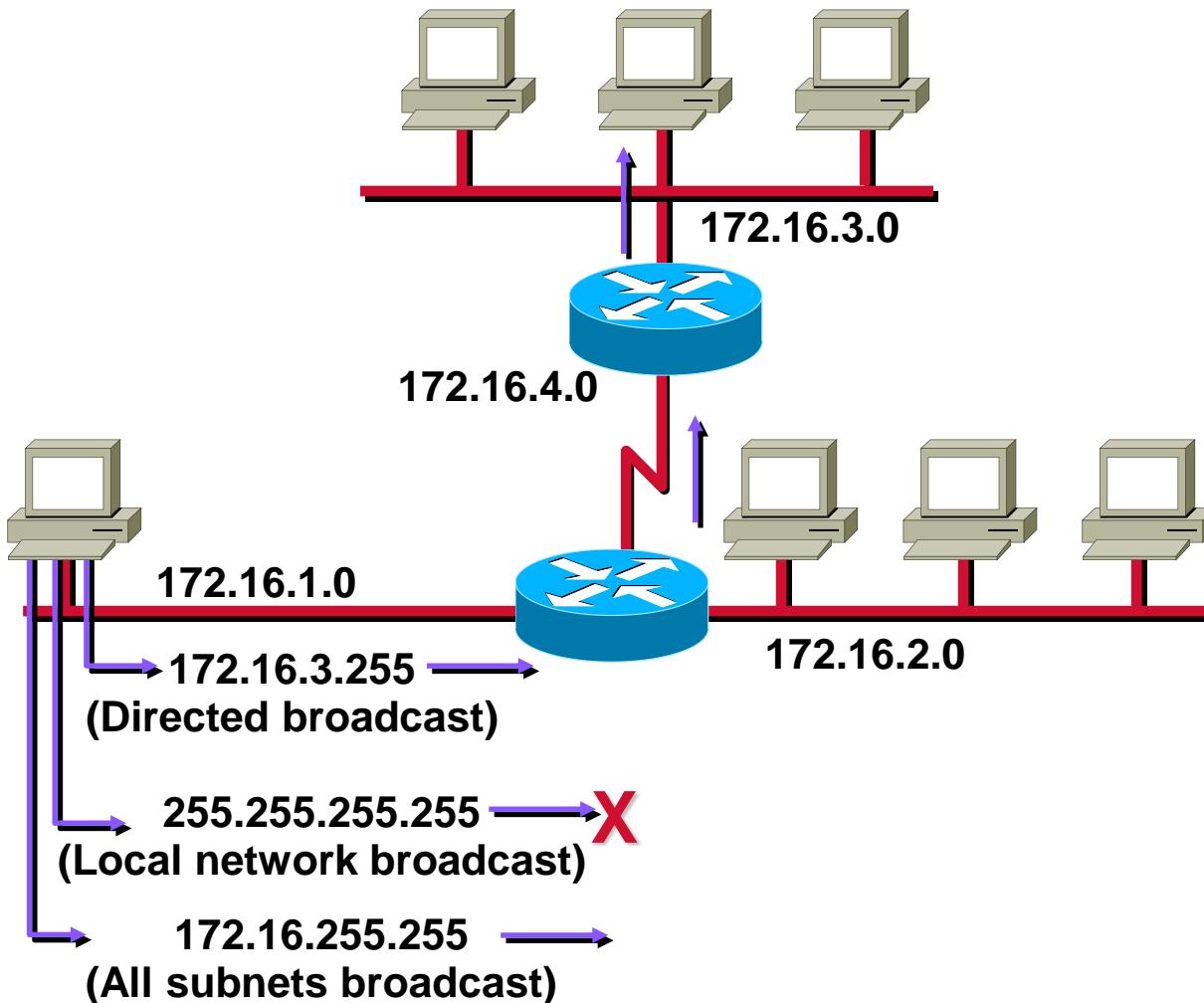
Subnet Mask Exercise

Address	Subnet Mask	Class	Subnet
172.16.2.10	255.255.255.0		
10.6.24.20	255.255.240.0		
10.30.36.12	255.255.255.0		

Subnet Mask Exercise Answers

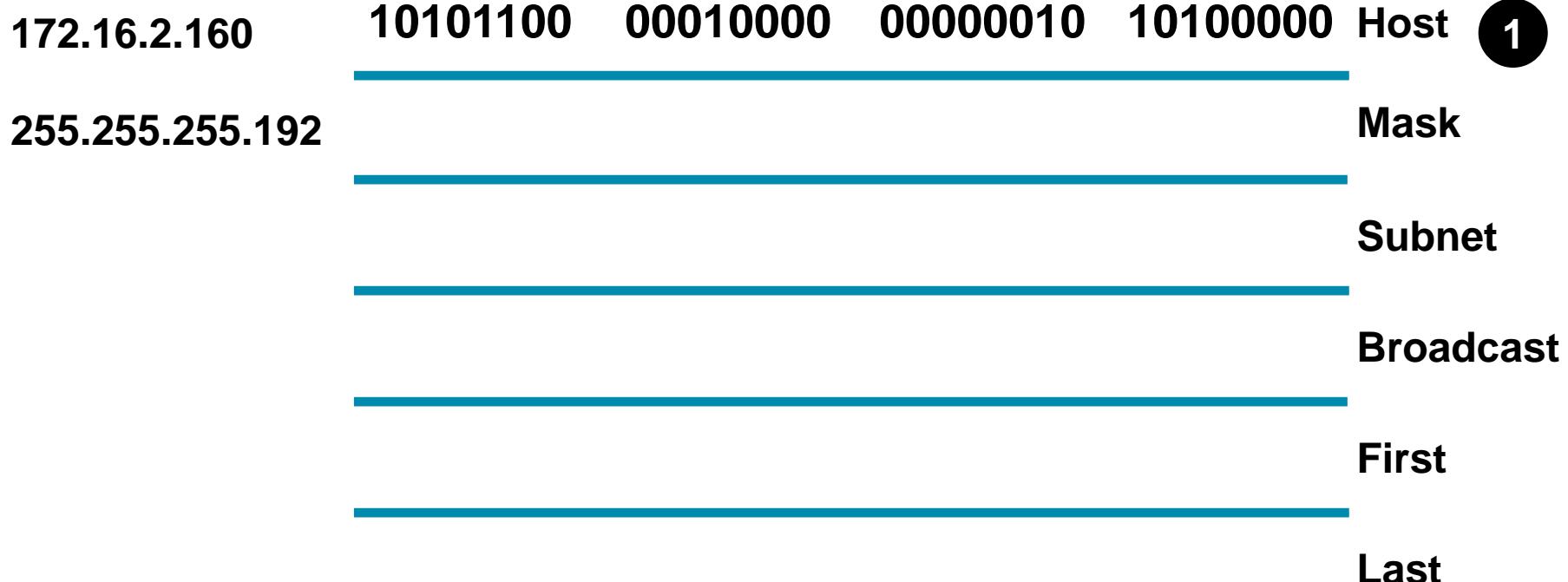
Address	Subnet Mask	Class	Subnet
172.16.2.10	255.255.255.0	B	172.16.2.0
10.6.24.20	255.255.240.0	A	10.6.16.0
10.30.36.12	255.255.255.0	A	10.30.36.0

Broadcast Addresses



Addressing Summary Example

172	16	2	160
-----	----	---	-----



Addressing Summary Example

172	16	2	160
-----	----	---	-----

172.16.2.160

10101100 00010000 00000010 10100000 Host 1

255.255.255.192

11111111 11111111 11111111 11000000 Mask 2

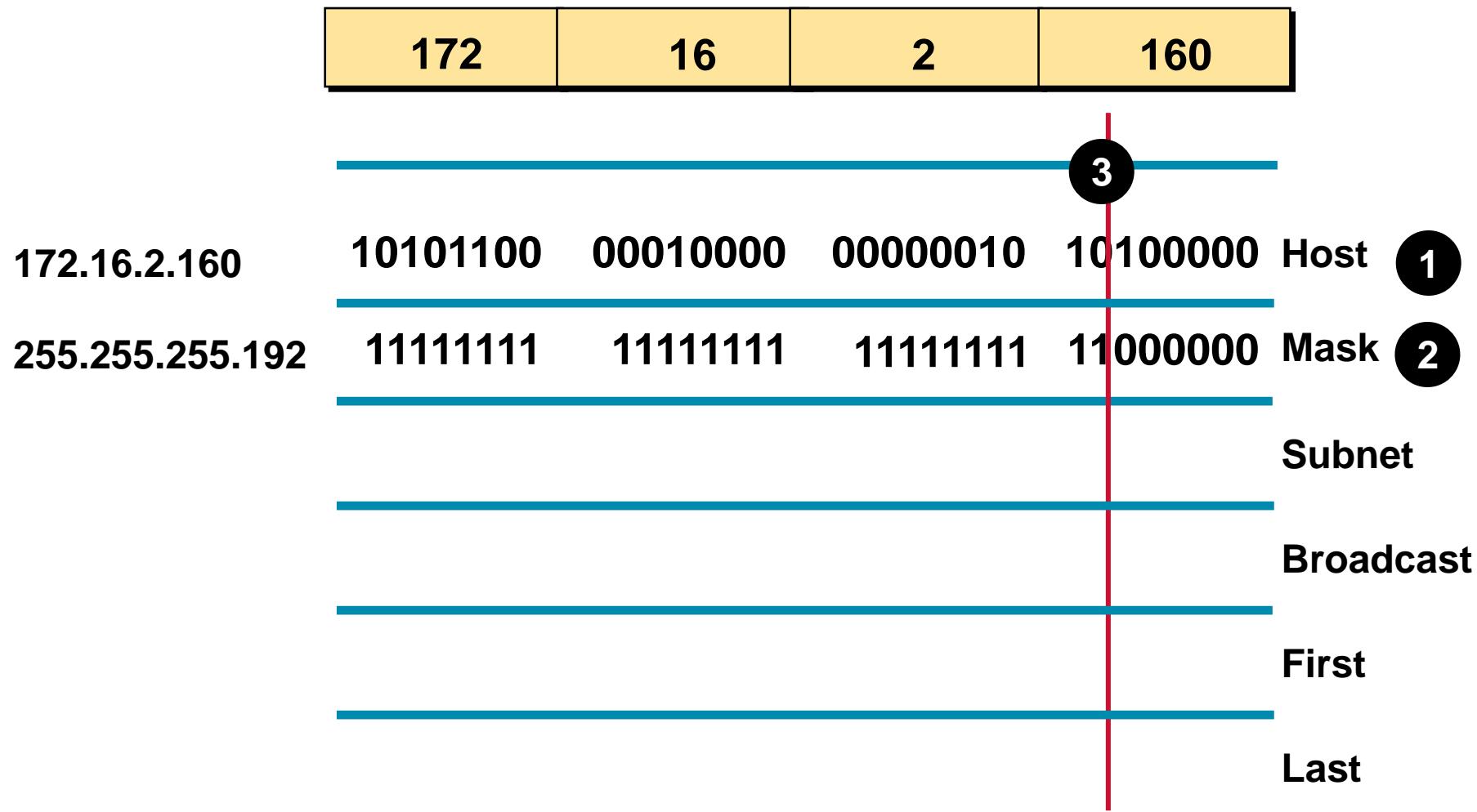
Subnet

Broadcast

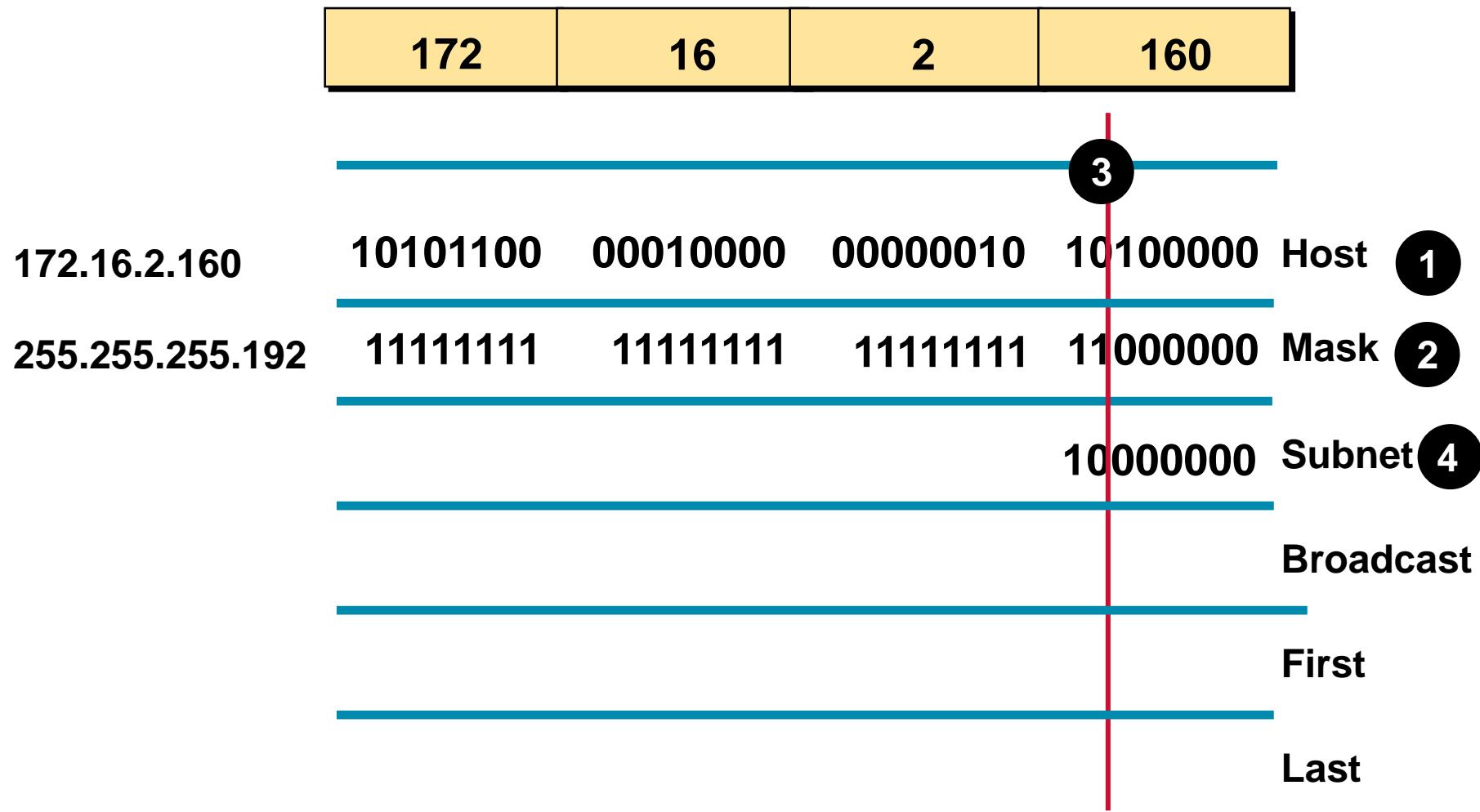
First

Last

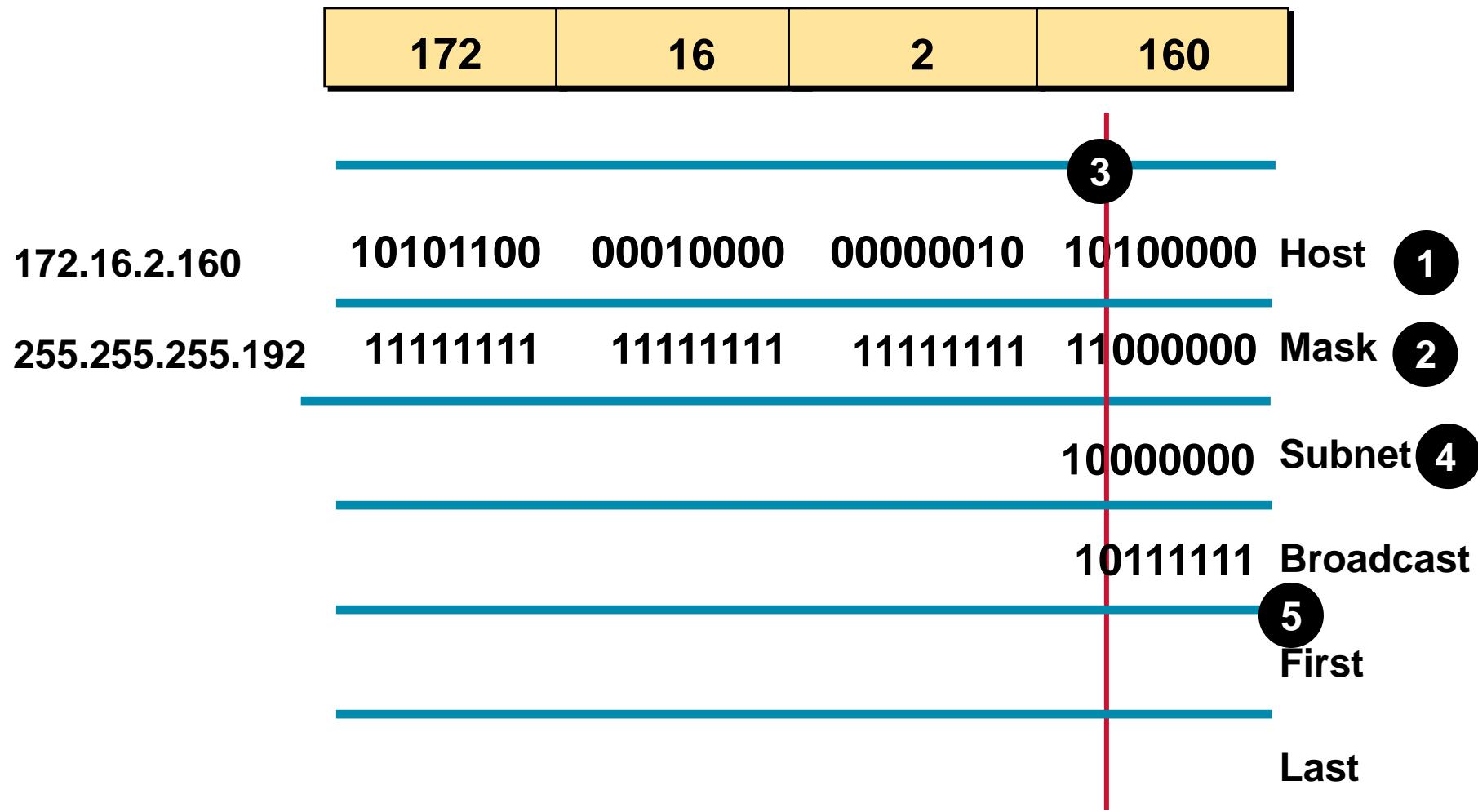
Addressing Summary Example



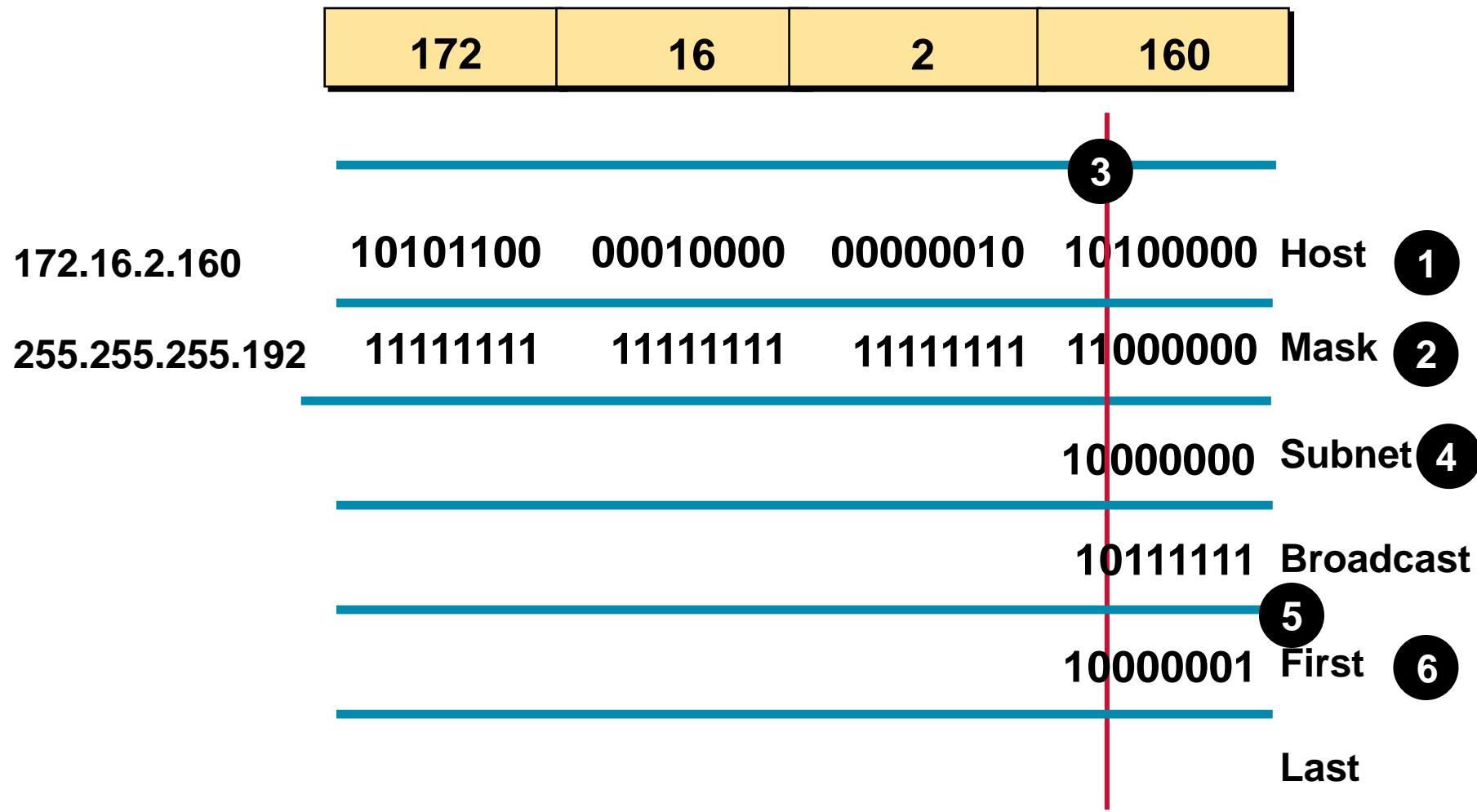
Addressing Summary Example



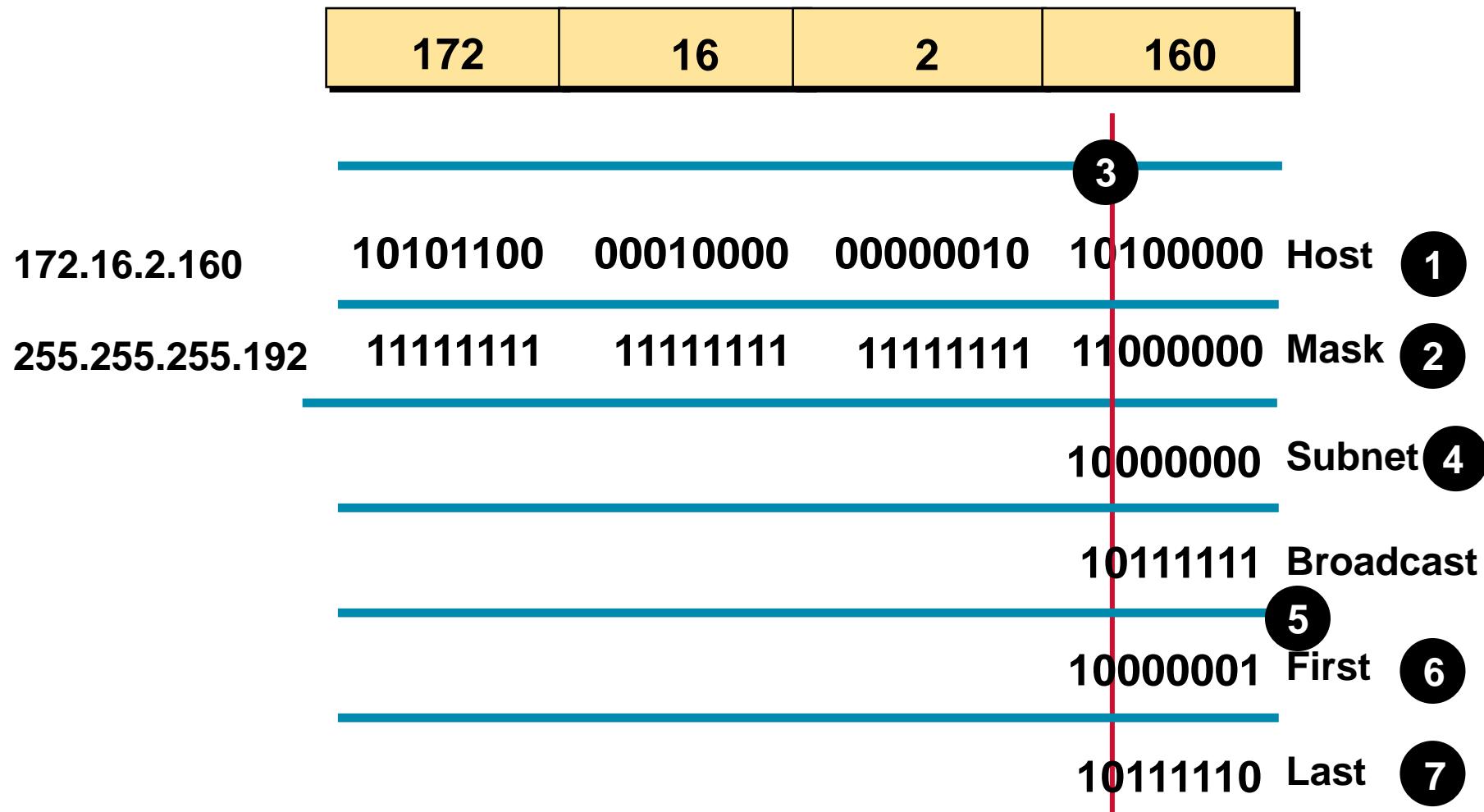
Addressing Summary Example



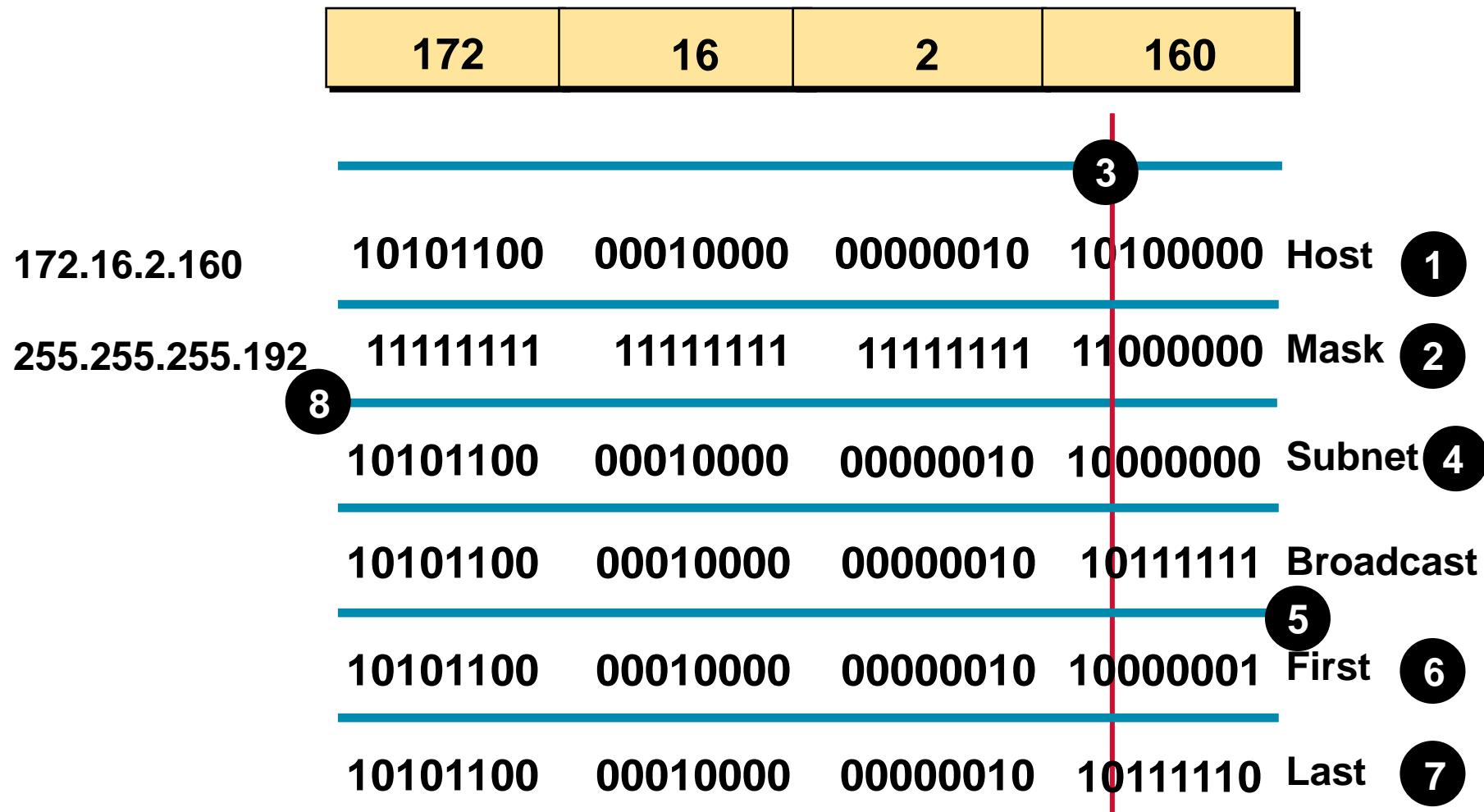
Addressing Summary Example



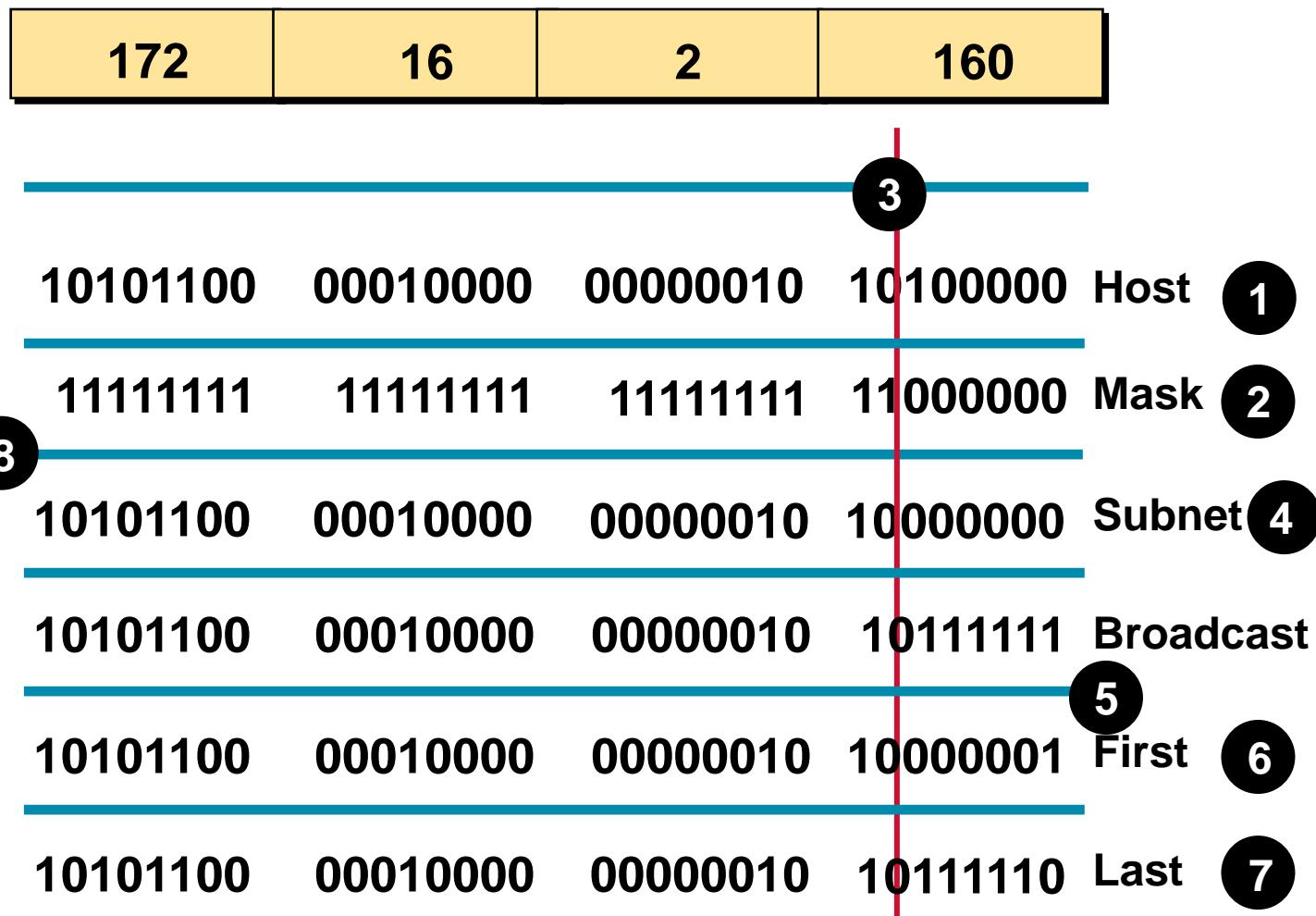
Addressing Summary Example



Addressing Summary Example



Addressing Summary Example



Class B Subnet Example

IP Host Address: 172.16.2.121

Subnet Mask: 255.255.255.0

Network	Network	Subnet	Host
172.16.2.121: 10101100	00010000	00000010	01111001
255.255.255.0: 11111111	11111111	11111111	00000000
Subnet: 10101100	00010000	00000010	00000000
Broadcast: 10101100	00010000	00000010	11111111

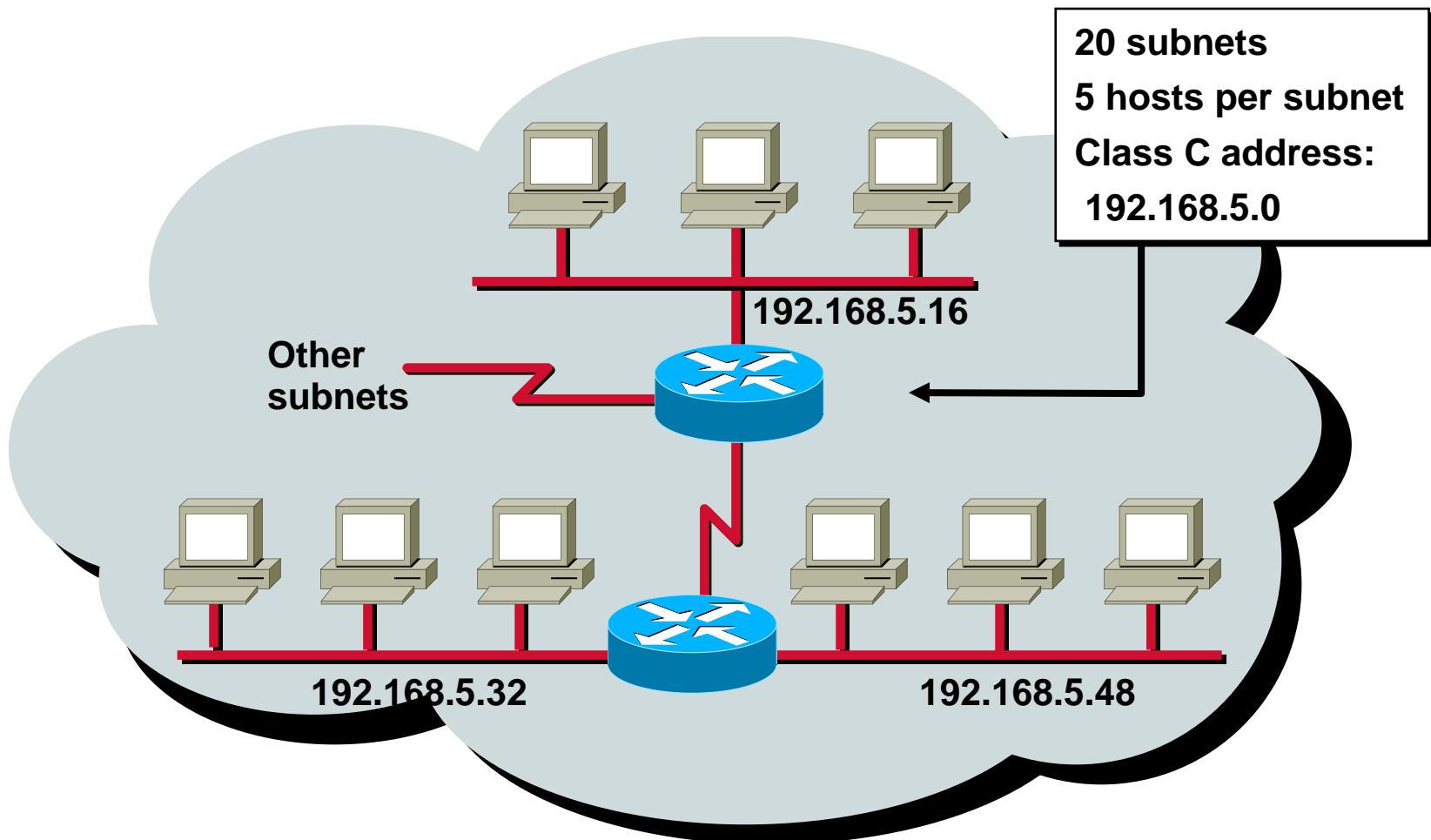
Subnet Address = 172.16.2.0

Host Addresses = 172.16.2.1–172.16.2.254

Broadcast Address = 172.16.2.255

Eight bits of subnetting

Subnet Planning



Class C Subnet Planning Example

IP Host Address: 192.168.5.121

Subnet Mask: 255.255.255.248

Network	Network	Network	Subnet	Host
192.168.5.121: 11000000	10101000	00000101	01111001	
255.255.255.248: 11111111	11111111	11111111	11111000	
Subnet:	11000000	10101000	00000101	01111000
Broadcast:	11000000	10101000	00000101	01111111

Subnet Address = 192.168.5.120

Host Addresses = 192.168.5.121–192.168.5.126

Broadcast Address = 192.168.5.127

Five Bits of Subnetting

Broadcast Addresses Exercise

Address	Subnet Mask	Class	Subnet	Broadcast
201.222.10.60	255.255.255.248			
15.16.193.6	255.255.248.0			
128.16.32.13	255.255.255.252			
153.50.6.27	255.255.255.128			

Broadcast Addresses Exercise Answers

Address	Subnet Mask	Class	Subnet	Broadcast
201.222.10.60	255.255.255.248	C	201.222.10.56	201.222.10.63
15.16.193.6	255.255.248.0	A	15.16.192.0	15.16.199.255
128.16.32.13	255.255.255.252	B	128.16.32.12	128.16.32.15
153.50.6.27	255.255.255.128	B	153.50.6.0	153.50.6.127

Firewalls

Presented By
Santosh Kumar

Outline

- Introduction
- Firewall Environments
- Type of Firewalls
- Future of Firewalls
- Conclusion

Introduction

- Firewalls control the flow of network traffic
- Firewalls have applicability in networks where there is no internet connectivity
- Firewalls operate on number of layers
- Can also act as VPN gateways
- Active content filtering technologies

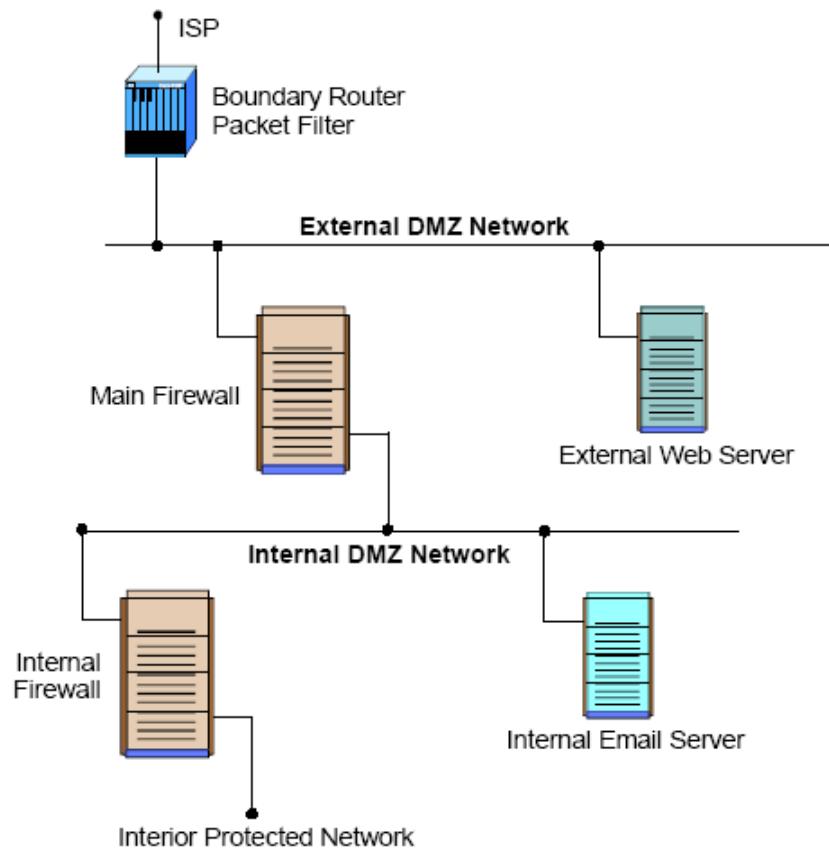
Firewall Environments

- There are different types of environments where a firewall can be implemented.
- Simple environment can be a packet filter firewall
- Complex environments can be several firewalls and proxies

DMZ Environment

- Can be created out of a network connecting two firewalls
- Boundary router filter packets protecting server
- First firewall provide access control and protection from server if they are hacked

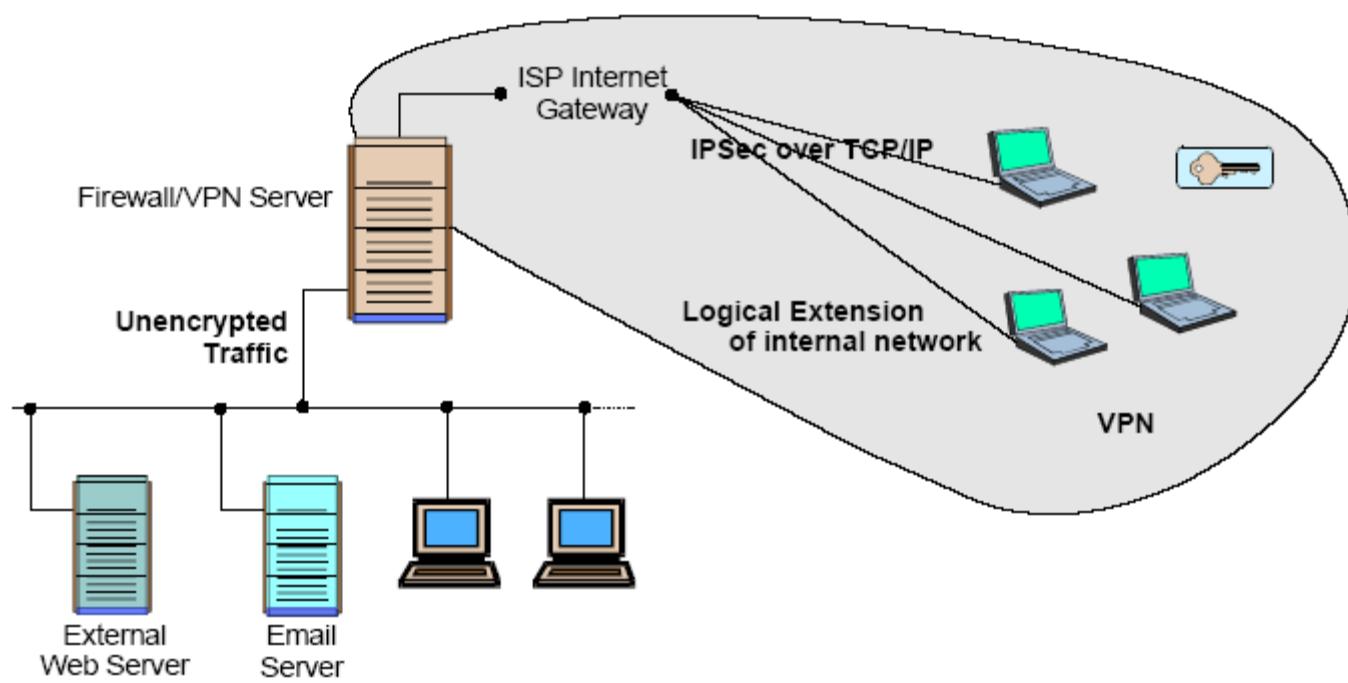
DMZ ENV



VPN

- VPN is used to provide secure network links across networks
- VPN is constructed on top of existing network media and protocols
- On protocol level IPsec is the first choice
- Other protocols are PPTP, L2TP

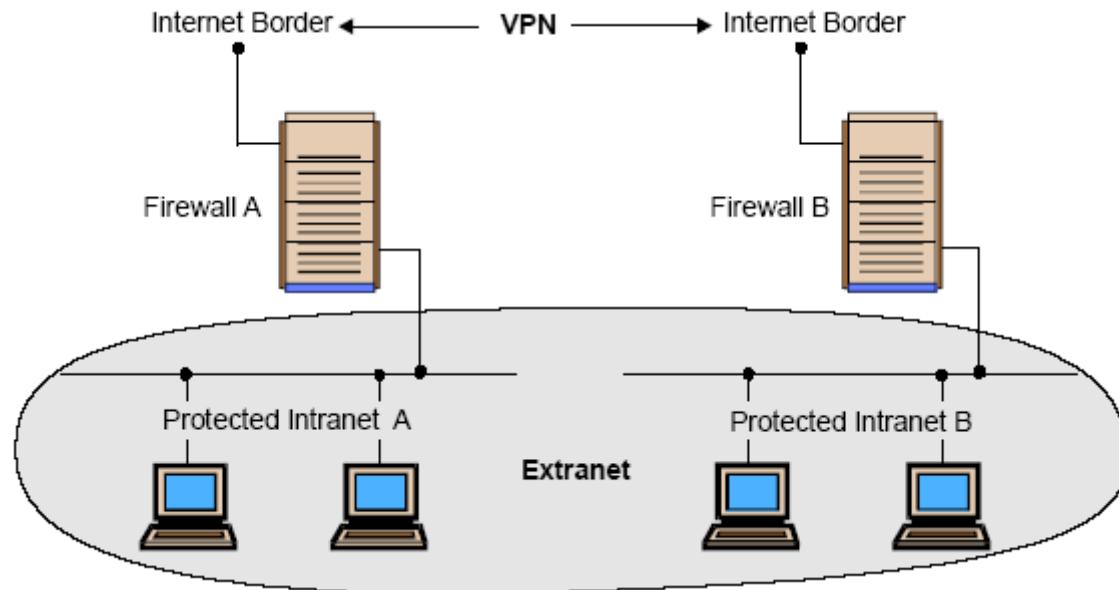
VPN



Intranets

- An intranet is a network that employs the same types of services, applications, and protocols present in an Internet implementation, without involving external connectivity
- Intranets are typically implemented behind firewall environments.

Intranets



Extranets

- Extranet is usually a business-to-business intranet
- Controlled access to remote users via some form of authentication and encryption such as provided by a VPN
- Extranets employ TCP/IP protocols, along with the same standard applications and services

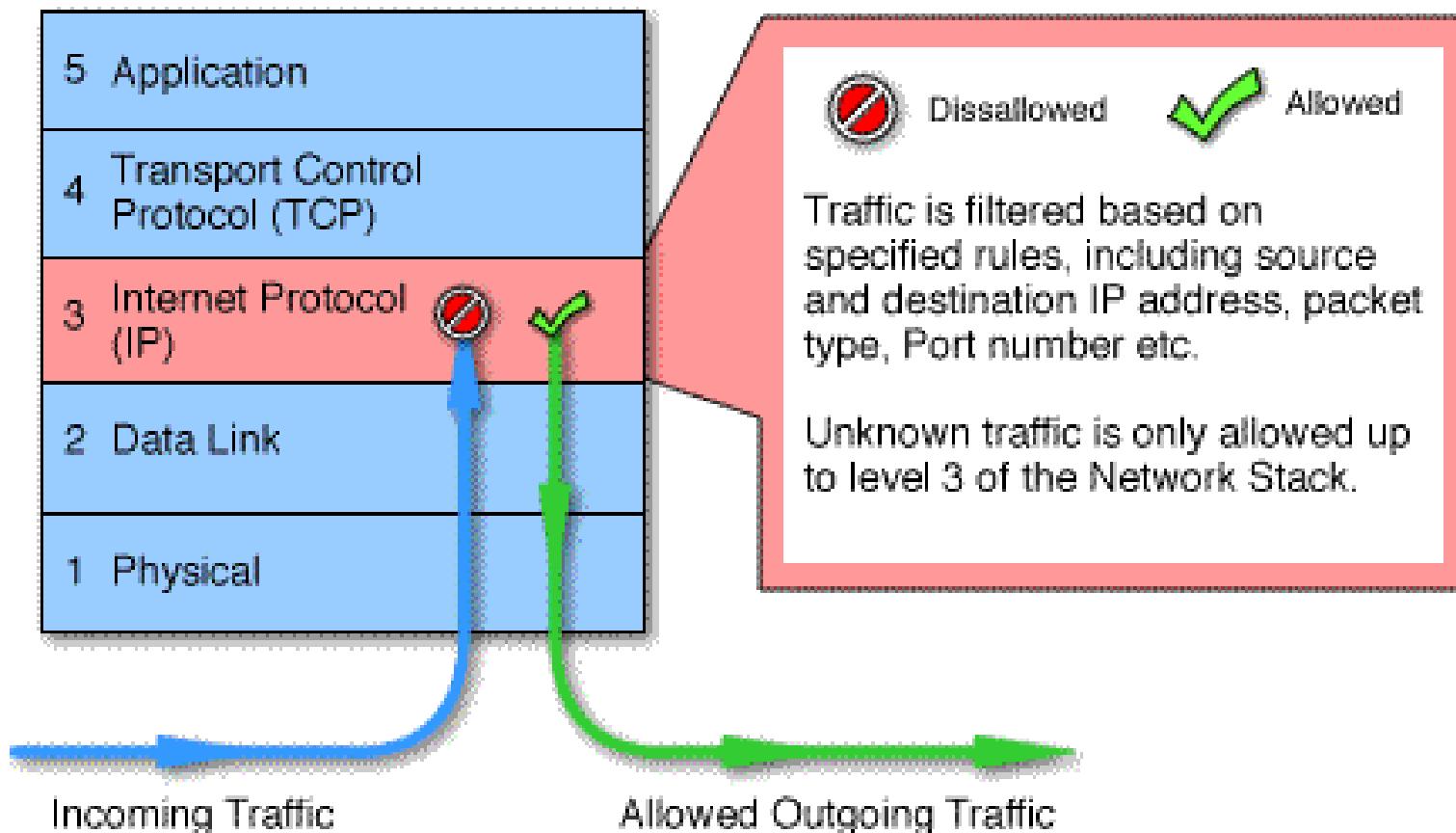
Type is Firewalls

- Firewalls fall into four broad categories
- Packet filters
- Circuit level
- Application level
- Stateful multilayer

Packet Filter

- Work at the network level of the OSI model
- Each packet is compared to a set of criteria before it is forwarded
- Packet filtering firewalls is low cost and low impact on network performance

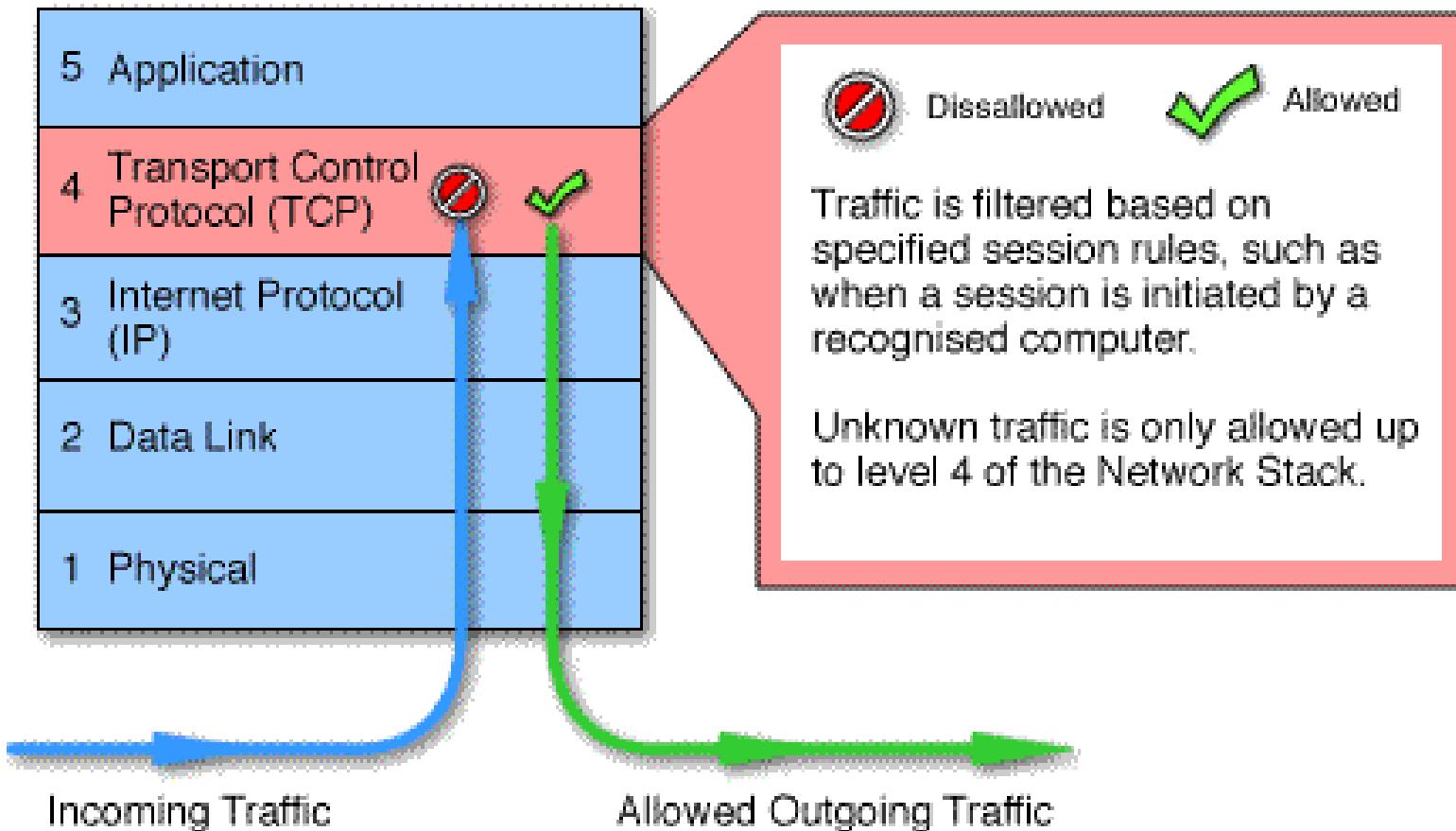
Packet Filtering



Circuit level

- Circuit level gateways work at the session layer of the OSI model, or the TCP layer of TCP/IP
- Monitor TCP handshaking between packets to determine whether a requested session is legitimate.

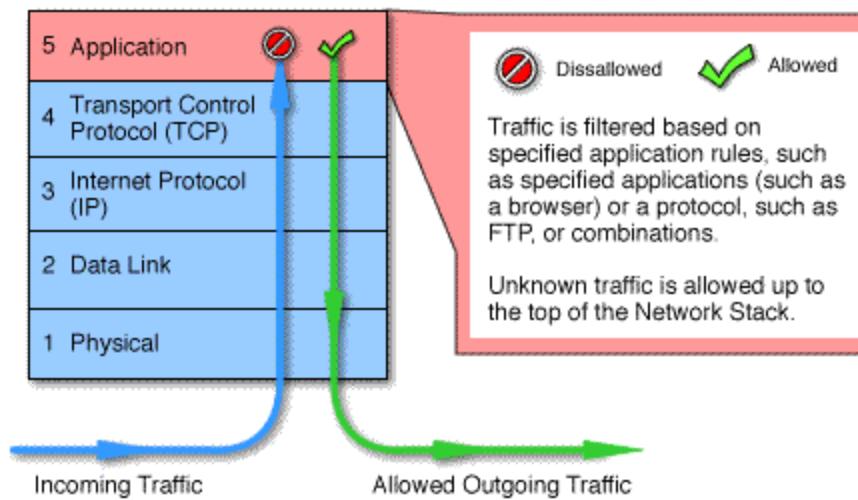
Circuit Level



Application Level

- Application level gateways, also called proxies, are similar to circuit-level gateways except that they are application specific
- Gateway that is configured to be a web proxy will not allow any ftp, gopher, telnet or other traffic through

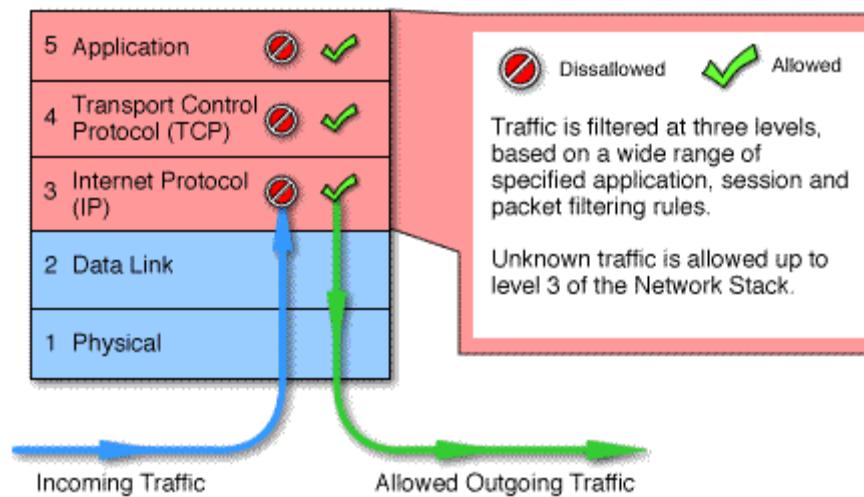
Application Level



Stateful Multilayer

- Stateful multilayer inspection firewalls combine the aspects of the other three types of firewalls
- They filter packets at the network layer, determine whether session packets are legitimate and evaluate contents of packets at the application layer

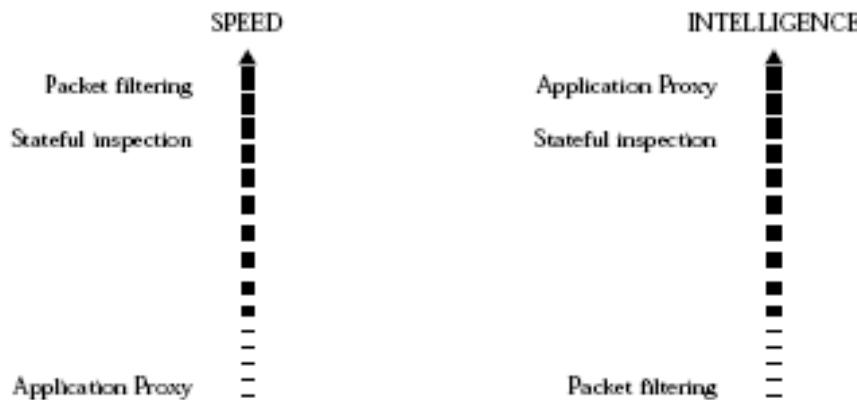
Stateful Multilayer



General Performance

FIREWALL PERFORMANCE SUMMARY

Technology	Speed	Flexibility	Intelligence
Packet filtering	V. Good	V.Good	Low
Application Proxy	Low	Low	V. Good
Stateful inspection	Good	Good	Good
Circuit gateway	Low	Low	Low



Future of Firewalls

- Firewalls will continue to advance as the attacks on IT infrastructure become more and more sophisticated
- More and more client and server applications are coming with native support for proxied environments
- Firewalls that scan for viruses as they enter the network and several firms are currently exploring this idea, but it is not yet in wide use

Conclusion

- It is clear that some form of security for private networks connected to the Internet is essential
- A firewall is an important and necessary part of that security, but cannot be expected to perform all the required security functions.

History of Linux

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system. It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers. Linux also runs on embedded systems, which are devices whose operating system is typically built into the firmware and is highly tailored to the system; this includes mobile phones, tablet computers, network routers, facility automation controls, televisions and video game consoles. Android, which is a widely used operating system for mobile devices, is built on top of the Linux kernel.

The Unix operating system was conceived and implemented in 1969 at AT&T's Bell Laboratories in the United States. The Unix operating system was conceived and implemented in 1969 at AT&T's Bell Laboratories in the United States.

The GNU Project is a free software, mass collaboration project, announced on 27 September 1983, by Richard Stallman at MIT. Its aim is to give computer users freedom and control in their use of their computers and computing devices, by collaboratively developing and providing software that is based on the following freedom rights: users are free to run the software, share it (copy, distribute), study it and modify it. GNU software guarantees these freedom-rights legally (via its license), and is therefore free software; the use of the word "free" always being taken to refer to freedom.

The GNU Project, started in 1983 by Richard Stallman, had the goal of creating a "complete Unix-compatible software system" composed entirely of free software. Work began in 1984. Later, in 1985, Stallman started the Free Software Foundation and wrote the GNU General Public License (GNU GPL) in 1989. By the early 1990s, many of the programs required in an operating system (such as libraries, compilers, text editors, a UNIX shell, and a windowing system) were completed, although low-level elements such as device drivers, daemons, and the kernel were stalled and incomplete.

MINIX, initially released in 1987, is an inexpensive minimal Unix-like operating system, designed for education in computer science, written by Andrew S. Tanenbaum. MINIX became free and was redesigned for use in embedded systems.

In 1991, while attending the University of Helsinki, Linus Torvalds became curious about operating systems. He began to work on his own operating system which eventually became the Linux kernel.

Torvalds began the development of the Linux kernel on MINIX and applications written for MINIX were also used on Linux. Later, Linux matured and further Linux kernel development took place on Linux systems. GNU applications also replaced all MINIX components, because it was advantageous to use the freely available code from the GNU Project with the fledgling operating system; code licensed under the GNU GPL can be reused in other projects as long as they also are released under the same or a compatible license. Torvalds initiated a switch from his original license, which prohibited commercial redistribution, to the GNU GPL. Developers worked to integrate GNU components with the Linux kernel, making a fully functional and free operating system.

The Name Linux & Logo Tux

Torvalds first named his OS *freakx*, combining "free", "reak" & "unix". As developments became rapid, he uploaded it into an FTP server at his University itself. His co-worker and the FTP server administrator thought the name *freakx* is not good for further developments. So he renamed it *Linux*, combining the name of "Linus" himself & Unix, without the consent of Linus. Later on Torvalds agreed to it & thus the name *Linux*.

One fine day, Linus visited an aquarium in Australia where some lovely penguins were allowed to wander around. As Linus came close to one, it bit him but also was looking very innocent. Consequently Linus decided to use the Penguin TUX as the trademark logo of Linux.

Linux Flavours

Although there are a large number of Linux implementations, you will find a lot of similarities in the different distributions, if only because every Linux machine is a box with building blocks that you may put together following your own needs and views. Linux may appear different depending on the distribution, your hardware and personal taste, but the fundamentals on which all graphical and other interfaces are built, remain the same. The Linux system is based on GNU tools (Gnu's Not UNIX), which provide a set of standard ways to handle and use the system. All GNU tools are open source, so they can be installed on any system. Most distributions offer pre-compiled packages of most common tools, such as RPM packages on RedHat and Debian packages (also called deb or dpkg) on Debian. However, if you are and like doing things yourself, you will enjoy Linux all the better, since most distributions come with a complete set of development tools, allowing installation of new software purely from source code. This setup also allows you to install software even if it does not exist in a pre-packaged form suitable for your system.

List of Main Distros & Release Years

Flavour	Year
Linux Kernel 1.0	1991
Slackware	1993
Debian	1993
Suse	1994
RedHat	1994
Crux	2001
Gentoo	2002
Puppy	2002

Linux Console

The Linux console is a system console internal to the Linux kernel (a system console is the device which receives all kernel messages and warnings and which allows logins in single user mode). The Linux console provides a way for the kernel and other processes to send text output to the user, and to receive text input from the user. The user typically enters text with a computer keyboard and reads the output text on a computer monitor. The Linux kernel supports virtual consoles - consoles that are logically separate, but which access the same physical keyboard and display. The Linux console was one of the first features of the kernel and was originally written by Linus Torvalds in 1991.



UNC
INFORMATION
TECHNOLOGY SERVICES

Shell Scripting

Santosh Kumar
Assistant Professor
AI&DS Department, VIIT, Pune
santosh.kumar@viit.ac.in

- Introduction
 - UNIX/LINUX and Shell
 - UNIX Commands and Utilities
 - Basic Shell Scripting Structure
- Shell Programming
 - Variable
 - Operators
 - Logic Structures
- Examples of Application in Cloud & Devops
- Hands-on Exercises

Why Shell Scripting ?

- Shell scripts can be used to prepare input files, job monitoring, and output processing.
- Useful to create own commands.
- Save lots of time on file processing.
- To automate some task of day to day life.
- System Administration part can be also automated.

Objectives & Prerequisites

- **After this workshop, you should be:**
 - Familiar with UNIX/LINUX, Borne Shell, shell variables/operators
 - Able to write simple shell scripts to illustrate programming logic
 - Able to write scripts for Cloud computing purposes
- **We assume that you have/know**
 - An account on the Emerald cluster
 - Basic knowledge of UNIX/LINUX and commands
 - UNIX editor e.g. vi or emacs

History of UNIX/Linux

- Unix is a command line operating system developed around 1969 in the Bell Labs
- Originally written using C
- Unix is designed so that users can extend the functionality
 - To build new tools easily and efficiently
 - To customize the shell and user interface.
 - To string together a series of Unix commands to create new functionality.
 - To create custom commands that do exactly what we want.
- Around 1990 Linus Torvalds of Helsinki University started off a freely available academic version of Unix
- Linux is the Antidote to a Microsoft dominated future

What is UNIX/Linux ?

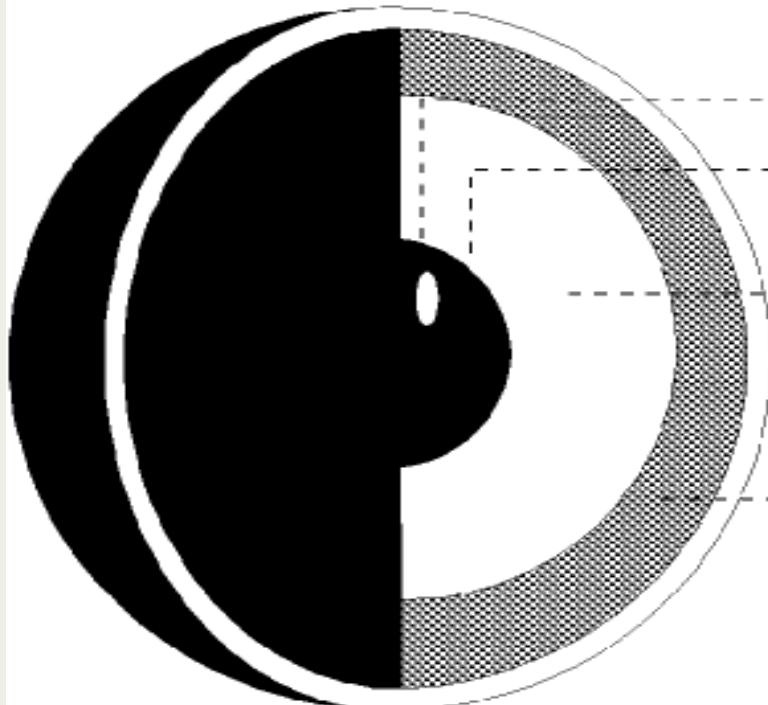
Simply put

- Multi-Tasking O/S
- Multi-User O/S
- Available on a range of Computers

■ SunOS	Sun Microsystems
■ IRIX	Silicon Graphics
■ HP-UX	Hewlett Packard
■ AIX	IBM
■ Linux



UNIX/LINUX Architecture



CPU
KERNEL
Resource allocation

SHELL
Command language interpreter
Interface to other OS parts

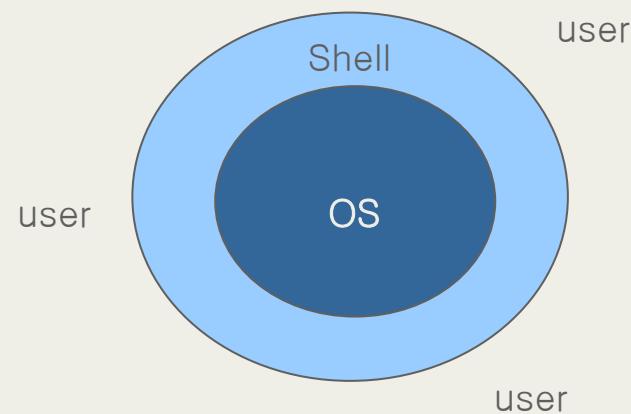
PROGRAMS
List Processors
Text processors
Programmer's tools
Programming languages
Networks and communications

APPLICATIONS

What is a “Shell”?

- The “Shell” is simply *another program* on top of the kernel which provides a basic human-OS interface.
 - It is a command interpreter
 - ◆ Built on top of the kernel
 - ◆ Enables users to run services provided by the UNIX OS
 - In its simplest form, a series of commands in a file is a shell program that saves having to retype commands to perform common tasks.
- How to know what shell you use

```
echo $SHELL
```



- sh Bourne Shell (Original Shell) (*Steven Bourne of AT&T*)
- bash Bourne Again Shell (*GNU Improved Bourne Shell*)
- csh C-Shell (C-like Syntax)(*Bill Joy of Univ. of California*)
- ksh Korn-Shell (Bourne+some C-shell)(*David Korn of AT&T*)
- tcsh Turbo C-Shell (More User Friendly C-Shell).
- To check shell:
 - \$ echo \$SHELL (shell is a pre-defined variable)
- To switch shell:
 - \$ exec shellname (e.g., \$ exec bash or simply type \$ bash)
 - You can switch from one shell to another by just typing the name of the shell. `exit` return you back to previous shell.

Which Shell to Use?

- **sh** (Bourne shell) was considered better for programming
- **csh** (C-Shell) was considered better for interactive work.
- **tcsh** and **korn** were improvements on c-shell and bourne shell respectively.
- **bash** is largely compatible with sh and also has many of the nice features of the other shells
- On many systems such as our LINUX clusters sh is symbolically linked to bash, /bin/sh -> /bin/bash
- We recommend that you use sh/bash for writing new shell scripts but learn csh/tcsh to understand existing scripts.
- Many, if not all, scientific applications require csh/tcsh environment (GUI, Graphics Utility Interface)
- **All Linux versions use the Bash shell (Bourne Again Shell) as the default shell**
 - Bash/Bourn/ksh/sh prompt: \$
- **All UNIX system include C shell and its predecessor Bourne shell.**
 - Csh/tcsh prompt: %

What is Shell Script?

- A **shell script** is a script written for the shell
- Two key ingredients
 - UNIX/LINUX commands
 - Shell programming syntax

A Shell Script Example

```
#!/bin/sh

`ls -l *.log| awk '{print $8}' |sed 's/.log//g' > file_list` 

cat file_list|while read each_file
do
    babel -ig03 $each_file".log" -oxyz $each_file".xyz"

    echo '# nosymmetry integral=Grid=UltraFine scf=tight rhf/6-311++g** pop=(nbo,chelpg)'>head
    echo '' >>head
    echo "$each_file" opt pop nbo chelp aim charges '>> head
    echo '' >>head
    echo '0 1 '>>head

    `sed '1,2d' $each_file.xyz >junk`
    input=./$each_file".com"
    cat head > $input
    cat junk >> $input
    echo '' >> $input

done
/bin/rm ./junk ./head ./file_list
```

UNIX/LINUX Commands

- File Management and Viewing
- Filesystem Management
- Help,Job/Process Management
- Network Management
- System Management
- User Management
- Printing and Programming
- Document Preparation
- Miscellaneous
- To understand the working of the command and possible options use ([man](#) command)
- Using the GNU Info System ([info](#), info command)
- Listing a Description of a Program ([whatis](#) command)
- Many tools have a long-style option, `--help', that outputs usage information about the tool, including the options and arguments the tool takes. Ex: `whoami --help`

File and Directory Management

- **cd** Change the current directory. With no arguments "cd" changes to the users home directory. (cd <directory path>)
- **chmod** Change the file permissions.

Ex: chmod 751 myfile : change the file permissions to rwx for owner, rx for group and x for others (**x=1,r=4,w=2**)

Ex: chmod go=+r myfile : Add read permission for the group and others (character meanings u-user, g-group, o-other, + add permission,-remove,r-read,w-write,x-exe)

Ex: chmod +s myfile - Setuid bit on the file which allows the program to run with user or group privileges of the file.

- **chown** Change owner.

Ex: chown <owner1> <filename> : Change ownership of a file to owner1.

- **chgrp** Change group.

Ex: chgrp <group1> <filename> : Change group of a file to group1.

- **cp** Copy a file from one location to another.

Ex: cp file1 file2 : Copy file1 to file2; Ex: cp –R dir1 dir2 : Copy dir1 to dir2

File and Directory Management

■ **ls** List contents of a directory.

Ex: ls, ls -l , ls -al, ls -ld, ls -R

■ **mkdir** Make a directory.

Ex: mkdir <directory name> : Makes a directory

Ex *mkdir -p /www/chache/var/log* will create all the directories starting from www.

■ **mv** Move or rename a file or directory.

Ex: mv <source> <destination>

■ **find** Find files (find <start directory> -name <file name> -print)

Ex: *find /home -name readme -print*

Search for readme starting at home and output full path, "/home" = Search starting at the home directory and proceed through all its subdirectories; "-name readme" = Search for a file named readme "-print" = Output the full path to that file

■ **locate** File locating program that uses the slocate database.

Ex: locate -u to create the database,

locate <file/directory> to find file/directory

- **pwd** Print or list the present working directory with full path.
- **rm** Delete files (Remove files). (rm –rf <directory/file>)
- **rmdir** Remove a directory. The directory must be empty. (rmdir <directory>)
- **touch** Change file timestamps to the current time. Make the file if it doesn't exist. (touch <filename>)
- **whereis** Locate the binary and man page files for a command. (whereis <program/command>)
- **which** Show full path of commands where given commands reside. (which <command>)

File viewing and editing

- **emacs** Full screen editor.
- **pico** Simple text editor.
- **vi** Editor with a command mode and text mode. Starts in command mode.
- **gedit** GUI Text Editor
- **tail** Look at the last 10 lines of a file.

Ex: tail –f <filename> ; Ex: tail -100 <filename>

- **head** Look at the first 10 lines of a file. (head <filename>)

File and Directory Management

File compression, backing up and restoring

- **compress** Compress data.
- **uncompress** Expand data.
- **cpio** Can store files on tapes. to/from archives.
- **gzip** - zip a file to a gz file.
- **gunzip** - unzip a gz file.
- **tar** Archives files and directories. Can store files and directories on tapes.

Ex: tar -zcvf <destination> <files/directories> - Archive copy groups of files. tar –zxvf <compressed file> to uncompress

- **zip** – Compresses a file to a .zip file.
- **unzip** – Uncompresses a file with .zip extension.
- **cat** View a file

Ex: cat filename

- **cmp** Compare two files.
- **cut** Remove sections from each line of files.

File and Directory Management

- **diff** Show the differences between files.
Ex: diff file1 file2 : Find differences between file1 & file2.
- **echo** Display a line of text.
- **grep** List all files with the specified expression.
(*grep pattern <filename/directorypath>*)
Ex: ls -l |grep sidbi : List all lines with a sidbi in them.
Ex: grep " R " : Search for R with a space on each side
- **sleep** Delay for a specified amount of time.
- **sort** Sort a file alphabetically.
- **uniq** Remove duplicate lines from a sorted file.
- **wc** Count lines, words, characters in a file. (wc –c/w/l <filename>).
- **sed** stream editor, extremely powerful!
- **awk** an extremely versatile programming language for working on files



- grep
 - Pattern searching
 - Example: `grep 'boo' filename`
- sed
 - Text editing
 - Example: `sed 's/XYZ/xyz/g' filename`
- awk
 - Pattern scanning and processing
 - Example: `awk '{print $4, $7}' filename`

Shell Scripting

- Start `vi scriptfilename.sh` with the line
`#!/bin/sh`
- All other lines starting with # are comments.
 - make code readable by including comments
- Tell Unix that the script file is executable
 - `$ chmod u+x scriptfilename.sh`
 - `$ chmod +x scriptfilename.sh`
- Execute the shell-script
 - `$./scriptfilename.sh`

My First Shell Script

```
$ vi myfirstscript.sh
```

```
#! /bin/sh
```

```
# The first example of a shell script
directory=`pwd`
echo Hello World!
echo The date today is `date`
echo The current directory is $directory
```

```
$ chmod +x myfirstscript.sh
```

```
$ ./myfirstscript.sh
```

```
Hello World!
```

```
The date today is Mon Mar 8 15:20:09 EST 2010
```

```
The current directory is /netscr/shubin/test
```

Shell Scripts

- **Text files that contain sequences of UNIX commands , created by a text editor**
- **No compiler required to run a shell script, because the UNIX shell acts as an **interpreter** when reading script files**
- **After you create a shell script, you simply tell the OS that the file is a program that can be executed, by using the **chmod** command to change the files' mode to be executable**
- **Shell programs run **less quickly** than compiled programs, because the shell must interpret each UNIX command inside the executable script file before it is executed**

Commenting

- Lines starting with # are comments except the very first line where # ! indicates the location of the shell that will be run to execute the script.
- On any line characters following an unquoted # are considered to be comments and ignored.
- Comments are used to;
 - Identify who wrote it and when
 - Identify input variables
 - Make code easy to read
 - Explain complex code sections
 - Version control tracking
 - Record modifications

Quote Characters

There are three different quote characters with different behaviour. These are:

- “ : **double quote**, weak quote. If a string is enclosed in “ ” the references to variables (i.e `$variable`) are replaced by their values. Also back-quote and escape \ characters are treated specially.
- ‘ : **single quote**, strong quote. Everything inside single quotes are taken literally, nothing is treated as special.
- ` : **back quote**. A string enclosed as such is treated as a command and the shell attempts to execute it. If the execution is successful the primary output from the command replaces the string.

Example: `echo "Today is:" `date``

Echo command is well appreciated when trying to debug scripts.

Syntax : echo {options} string

Options: -e : expand \ (back-slash) special characters

-n : do not output a new-line at the end.

String can be a “weakly quoted” or a ‘strongly quoted’ string. In the weakly quoted strings the references to variables are replaced by the value of those variables before the output.

As well as the variables some special backslash_escaped symbols are expanded during the output. If such expansions are required the -e option must be used.

User Input During Shell Script Execution

- As shown on the hello script input from the standard input location is done via the read command.
- Example

```
echo "Please enter three filenames:"  
read filea fileb filec  
echo "These files are used:$filea $fileb $filec"
```

- Each read statement reads an entire line. In the above example if there are less than 3 items in the response the trailing variables will be set to blank ‘ ’.
- Three items are separated by one space.

Hello script exercise continued...

- The following script asks the user to enter his name and displays a personalised hello.

```
#!/bin/sh  
  
echo "Who am I talking to?"  
  
read user_name  
  
echo "Hello $user_name"
```

- Try replacing “ with ‘ in the last line to see what happens.

Debugging your shell scripts

- Generous use of the `echo` command will help.
- Run script with the `-x` parameter.
E.g. `sh -x ./myscript`
or `set -o xtrace` before running the script.
- These options can be added to the first line of the script where the shell is defined.
e.g. `#!/bin/sh -xv`

Shell Programming

- **Programming features of the UNIX/LINUX shell:**
 - **Shell variables:** Your scripts often need to keep values in memory for later use. Shell variables are symbolic names that can access values stored in memory
 - **Operators:** Shell scripts support many operators, including those for performing mathematical operations
 - **Logic structures:** Shell scripts support **sequential logic** (for performing a series of commands), **decision logic** (for branching from one point in a script to another), **looping logic** (for repeating a command several times), and **case logic** (for choosing an action from several possible alternatives)

- **Variables are symbolic names that represent values stored in memory**
- **Three different types of variables**
 - **Global Variables:** Environment and configuration variables, capitalized, such as **HOME, PATH, SHELL, USERNAME, and PWD.**

When you login, there will be a large number of global System variables that are already defined. These can be freely referenced and used in your shell scripts.

- **Local Variables**

Within a shell script, you can create as many new variables as needed. Any variable created in this manner remains in existence only within that shell.

- **Special Variables**

Reversed for OS, shell programming, etc. such as positional parameters \$0, \$1 ...

A few global (environment) variables

SHELL	Current shell
DISPLAY	Used by X-Windows system to identify the display
HOME	Fully qualified name of your login directory
PATH	Search path for commands
MANPATH	Search path for <man> pages
PS1 & PS2	Primary and Secondary prompt strings
USER	Your login name
TERM	terminal type
PWD	Current working directory



Referencing Variables

Variable contents are accessed using '\$':

e.g. **\$ echo \$HOME**

\$ echo \$SHELL

To see a list of your environment variables:

\$ printenv

or:

\$ printenv | more

Defining Local Variables

- As in any other programming language, variables can be defined and used in shell scripts.
- Unlike other programming languages, variables in Shell Scripts are not typed.
- Examples :

a=1234 # a is NOT an integer, a string instead

b=\$a+1 # will not perform arithmetic but be the string '1234+1'

b=`expr \$a + 1` will perform arithmetic so b is 1235 now.

Note : +,-,/,*,**, % operators are available.

b=abcde # b is string

b='abcde' # same as above but much safer.

b=abc def # will not work unless 'quoted'

b='abc def' # i.e. this will work.

IMPORTANT NOTE: DO NOT LEAVE SPACES AROUND THE =

Referencing variables --curly bracket

- Having defined a variable, its contents can be referenced by the \$ symbol. E.g. \${variable} or simply \$variable. When ambiguity exists \$variable will not work. Use \${ } the rigorous form to be on the safe side.
- Example:

```
a='abc'
```

```
b=${a}def # this would not have worked without the{ } as  
#it would try to access a variable named adef
```

- To create lists (array) – round bracket

\$ set Y = (UNL 123 CS251)

- To set a list element – square bracket

\$ set Y[2] = HUSKER

- To view a list element:

\$ echo \$Y[2]

- Example:

```
#!/bin/sh
a=(1 2 3)
echo ${a[*]}
echo ${a[0]}
```

Results: 1 2 3

1

Positional Parameters

- When a shell script is invoked with a set of command line parameters each of these parameters are copied into special variables that can be accessed.
- **\$0** This variable that contains the name of the script
- **\$1, \$2, \$n** 1st, 2nd 3rd command line parameter
- **\$#** Number of command line parameters
- **\$\$** process ID of the shell
- **\$@** same as **\$*** but as a list one at a time (see for loops later)
- **\$?** Return code ‘exit code’ of the last command
- **Shift** command: This shell command shifts the positional parameters by one towards the beginning and drops \$1 from the list. After a shift \$2 becomes \$1 , and so on ... It is a useful command for processing the input parameters one at a time.

Example:

Invoke : ./myscript one two buckle my shoe

During the execution of **myscript** variables **\$1 \$2 \$3 \$4** and **\$5** will contain the values **one, two, buckle, my, shoe** respectively.

- **vi myinputs.sh**

```
#!/bin/sh
echo Total number of inputs: $#
echo First input: $1
echo Second input: $2
```
- **chmod u+x myinputs.sh**
- **myinputs.sh HUSKER UNL CSE**

```
Total number of inputs: 3
First input: HUSKER
Second input: UNL
```

- programming features of the UNIX shell:
 - ***Shell variables***
 - ***Operators***
 - ***Logic structures***

Shell Operators

- The Bash/Bourne/ksh shell operators are divided into three groups: **defining and evaluating operators**, **arithmetic operators**, and **redirecting and piping operators**

Defining and Evaluating

- A shell variable take on the generalized form **variable=value** (except in the C shell).

```
$ set x=37; echo $x
```

37

```
$ unset x; echo $x
```

x: Undefined variable.

- You can set a pathname or a command to a variable or substitute to set the variable.

```
$ set mydir=`pwd`; echo $mydir
```

Pipes & Redirecting

- **Piping:** An important early development in Unix , a way to pass the output of one tool to the input of another.

```
$ who | wc -l
```

By combining these two tools, giving the wc command the output of who, you can build a new command to **list the number of users currently on the system**

- **Redirecting via angle brackets:** Redirecting input and output follows a similar principle to that of piping except that redirects work with files, not commands.

```
tr '[a-z]' '[A-Z]' < $in_file > $out_file
```

The command must come first, the *in_file* is directed in by the less_than sign (<) and the *out_file* is pointed at by the greater_than sign (>).



Arithmetic Operators

- **expr supports the following operators:**
 - arithmetic operators: +,-,*/,%
 - comparison operators: <, <=, ==, !=, >=, >
 - boolean/logical operators: &, |
 - parentheses: (,)
 - precedence is the same as C, Java



- **vi math.sh**

```
#!/bin/sh
count=5
count=`expr $count + 1 `
echo $count
```

- **chmod u+x math.sh**
- **math.sh**

- **vi real.sh**

```
#!/bin/sh
a=5.48
b=10.32
c=`echo "scale=2; $a + $b" |bc`
echo $c
```

- **chmod u+x real.sh**

- **./real.sh**

15.80



Arithmetic operations in shell scripts

<code>var++ , var-- , ++var , --var</code>	post/pre increment/decrement
<code>+ , -</code>	add subtract
<code>* , / , %</code>	multiply/divide, remainder
<code>**</code>	power of
<code>! , ~</code>	logical/bitwise negation
<code>& , </code>	bitwise AND, OR
<code>&& </code>	logical AND, OR

Shell Programming

- programming features of the UNIX shell:
 - ***Shell variables***
 - ***Operators***
 - ***Logic structures***

Shell Logic Structures

The four basic logic structures needed for program development are:

- **Sequential logic:** to execute commands in the order in which they appear in the program
- **Decision logic:** to execute commands only if a certain condition is satisfied
- **Looping logic:** to repeat a series of commands for a given number of times
- **Case logic:** to replace “if then/else if/else” statements when making numerous comparisons



Conditional Statements (if constructs)

The most general form of the if construct is;

```
if command executes successfully
then
    execute command
elif this command executes successfully
then
    execute this command
    and execute this command
else
    execute default command
fi
```

However- elif and/or else clause can be omitted.

Examples

SIMPLE EXAMPLE:

```
if date | grep "Fri"
then
    echo "It's Friday!"
fi
```

FULL EXAMPLE:

```
if [ "$1" == "Monday" ]
then
    echo "The typed argument is Monday."
elif [ "$1" == "Tuesday" ]
then
    echo "Typed argument is Tuesday"
else
    echo "Typed argument is neither Monday nor Tuesday"
fi
```

Note: = or == will both work in the test but == is better for readability.

String and numeric comparisons used with test or [[]] which is an alias for test and also [] which is another acceptable syntax

- `string1 = string2` True if strings are identical
...ditto....
- `string1 !=string2` True if strings are not identical
- `string` Return 0 exit status (=true) if string is not null
- `-n string` Return 0 exit status (=true) if string is not null
- `-z string` Return 0 exit status (=true) if string is null

- `int1 -eq int2` Test identity
- `int1 -ne int2` Test inequality
- `int1 -lt int2` Less than
- `int1 -gt int2` Greater than
- `int1 -le int2` Less than or equal
- `int1 -ge int2` Greater than or equal

Combining tests with logical operators || (or) and && (and)

Syntax: if cond1 && cond2 || cond3 ...

An alternative form is to use a compound statement using the -a and -o keywords, i.e.

```
if cond1 -a cond2 -o cond3 ...
```

Where cond1,2,3 .. Are either commands returning a value or test conditions of the form [] or test ...

Examples:

```
if date | grep "Fri" && `date +'%H'` -gt 17
```

```
then
```

```
    echo "It's Friday, it's home time!!!"
```

```
fi
```

```
if [ "$a" -lt 0 -o "$a" -gt 100 ] # note the spaces around ] and [
```

```
then
```

```
    echo " limits exceeded"
```

```
fi
```

File enquiry operations

-d file	Test if file is a directory
-f file	Test if file is not a directory
-s file	Test if the file has non zero length
-r file	Test if the file is readable
-w file	Test if the file is writable
-x file	Test if the file is executable
-o file	Test if the file is owned by the user
-e file	Test if the file exists
-z file	Test if the file has zero length

All these conditions return true if satisfied and false otherwise.

■ A simple example

```
#!/bin/sh

if [ "$#" -ne 2 ] then
    echo $0 needs two parameters!
    echo You are inputting $# parameters.

else
    par1=$1
    par2=$2

fi

echo $par1
echo $par2
```

Another example:

```
#! /bin/sh
# number is positive, zero or negative
echo -e "enter a number:\c"
read number
if [ "$number" -lt 0 ]
then
    echo "negative"
elif [ "$number" -eq 0 ]
then
    echo zero
else
    echo positive
fi
```

Loop is a block of code that is repeated a number of times.

The repeating is performed either a pre-determined number of times determined by a list of items in the loop count (**for loops**) or until a particular condition is satisfied (**while** and **until loops**)

To provide flexibility to the loop constructs there are also two statements namely **break** and **continue** are provided.

for loops

Syntax:

```
for arg in list
do
    command(s)
...
done
```

Where the value of the variable *arg* is set to the values provided in the list one at a time and the block of statements executed. This is repeated until the list is exhausted.

Example:

```
for i in 3 2 5 7
do
    echo "$i times 5 is $(( i * 5 )) "
done
```

The while Loop

- A different pattern for looping is created using the **while** statement
- The **while** statement best illustrates how to set up a loop to test repeatedly for a matching condition
- The while loop tests an expression in a manner similar to the if statement
- As long as the statement inside the brackets is true, the statements inside the do and done statements repeat

while loops

Syntax:

```
while this_command_execute_successfully
do
    this command
    and this command
done
```

EXAMPLE:

```
while test "$i" -gt 0      # can also be while [ $i > 0 ]
do
    i=`expr $i - 1`
done
```

Looping Logic

- Example:

```
#!/bin/sh
for person in Bob Susan Joe Gerry
do
    echo Hello $person
done
```

Output:

```
Hello Bob
Hello Susan
Hello Joe
Hello Gerry
```

- Adding integers from 1 to 10

```
#!/bin/sh
i=1
sum=0
while [ "$i" -le 10 ]
do
    echo Adding $i into the sum.
    sum=`expr $sum + $i`
    i=`expr $i + 1`
done
echo The sum is $sum.
```

until loops

The syntax and usage is almost identical to the while-loops.

Except that the block is executed until the test condition is satisfied, which is the opposite of the effect of test condition in while loops.

Note: You can think of *until* as equivalent to *not_while*

Syntax:

```
until test
do
  commands ....
done
```

Switch/Case Logic

- The **switch logic structure simplifies the selection of a match when you have a list of choices**
- It **allows your program to perform one of many actions, depending upon the value of a variable**

Case statements

The case structure compares a string ‘usually contained in a variable’ to one or more patterns and executes a block of code associated with the matching pattern. Matching-tests start with the first pattern and the subsequent patterns are tested only if no match is not found so far.

case argument in

pattern 1) execute this command

and this

and this;;

pattern 2) execute this command

and this

and this;;

esac

Functions

- Functions are a way of grouping together commands so that they can later be executed via a single reference to their name. If the same set of instructions have to be repeated in more than one part of the code, this will save a lot of coding and also reduce possibility of typing errors.

SYNTAX:

```
functionname()  
{  
    block of commands  
}  
#!/bin/sh  
  
sum() {  
    x=`expr $1 + $2`  
    echo $x  
}
```

```
sum 5 3
```

```
echo "The sum of 4 and 7 is `sum 4 7`"
```

Take-Home Message

- **Shell script is a high-level language that must be converted into a low-level (machine) language by UNIX Shell before the computer can execute it**
- **UNIX shell scripts, created with the vi or other text editor, contain two key ingredients: a selection of UNIX commands glued together by Shell programming syntax**
- **UNIX/Linux shells are derived from the UNIX Bourne, Korn, and C/TCSH shells**
- **UNIX keeps three types of variables:**
 - Configuration; environmental; local
- **The shell supports numerous operators, including many for performing arithmetic operations**
- **The logic structures supported by the shell are sequential, decision, looping, and case**

To Script or Not to Script

■ Pros

- File processing
- Glue together compelling, customized testing utilities
- Create powerful, tailor-made manufacturing tools
- Cross-platform support
- Custom testing and debugging

■ Cons

- Performance slowdown
- Accurate scientific computing

Shell Scripting Examples

- Input file preparation
- Job submission
- Job monitoring
- Results processing

Input file preparation

```
#!/bin/sh

`ls -l *.log| awk '{print $8}' |sed 's/.log//g' > file_list`

cat file_list|while read each_file
do
    babel -ig03 $each_file".log" -oxyz $each_file".xyz"

    echo '# nosymmetry integral=Grid=UltraFine scf=tight rhf/6-311++g** pop=(nbo,chelpg)'>head
    echo '' >>head
    echo "$each_file" opt pop nbo chelp aim charges '>> head
    echo '' >>head
    echo '0 1 '>>head

    `sed '1,2d' $each_file.xyz >junk`
    input=./$each_file".com"
    cat head > $input
    cat junk >> $input
    echo '' >> $input

done
/bin/rm ./junk ./head ./file_list
```

LSF Job Submission

```
$ vi submission.sh
#!/bin/sh -f

#BSUB -q week
#BSUB -n 4
#BSUB -o output
#BSUB -J job_type
#BSUB -R "RH5 span[ptile=4]"
#BSUB -a mpichp4

mpirun.lsf ./executable.exe

exit
$chmod +x submission.sh
$bsub < submission.sh
```

Results Processing

```

#!/bin/sh
`ls -l *.out| awk '{print $8}'|sed 's/.out//g' > file_list`
cat file_list|while read each_file
do
    file1=./$each_file".out"
    Ts=`grep 'Kinetic energy =' $file1 |tail -n 1|awk '{print $4}' `
    Tw=`grep 'Total Steric Energy:' $file1 |tail -n 1|awk '{print $4}' `
    TsVne=`grep 'One electron energy =' $file1 |tail -n 1|awk '{print $5}' `
    Vnn=`grep 'Nuclear repulsion energy' $file1 |tail -n 1|awk '{print $5}' `
    J=`grep 'Coulomb energy =' $file1 |tail -n 1|awk '{print $4}' `
    Ex=`grep 'Exchange energy =' $file1 |tail -n 1|awk '{print $4}' `
    Ec=`grep 'Correlation energy =' $file1 |tail -n 1|awk '{print $4}' `
    Etot=`grep 'Total DFT energy =' $file1 |tail -n 1|awk '{print $5}' `
    HOMO=`grep 'Vector' $file1 | grep 'Occ=2.00'|tail -n 1|cut -c35-47|sed 's/D/E/g' `
    orb=`grep 'Vector' $file1 | grep 'Occ=2.00'|tail -n 1|awk '{print $2}' `
    orb=`expr $orb + 1 `
    LUMO=`grep 'Vector' $file1 |grep 'Occ=0.00'|grep '$orb' |tail -n 1|cut -c35-47|sed 's/D/E/g' `
    echo $each_file $Etot $Ts $Tw $TsVne $J $Vnn $Ex $Ec $HOMO $LUMO $steric >>out
done
/bin/rm file_list

```

Reference Books



- **Class Shell Scripting**
<http://oreilly.com/catalog/9780596005955/>
- **LINUX Shell Scripting With Bash**
<http://ebooks.ebookmall.com/title/linux-shell-scripting-with-bash-burtsch-ebooks.htm>
- **Shell Script in C Shell**
<http://www.grymoire.com/Unix/CshTop10.txt>
- **Linux Shell Scripting Tutorial**
<http://www.freeos.com/guides/lsst/>
- **Bash Shell Programming in Linux**
http://www.arachnoid.com/linux/shell_programming.html
- **Advanced Bash-Scripting Guide**
<http://tldp.org/LDP/abs/html/>
- **Unix Shell Programming**
<http://ebooks.ebookmall.com/title/unix-shell-programming-kochan-wood-ebooks.htm>



Questions & Comments

Please direct comments/questions about research computing to

E-mail: research@unc.edu

Please direct comments/questions pertaining to this presentation to

E-Mail: shubin@email.unc.edu

The PPT file of this presentation is available here:

http://its2.unc.edu/divisions/rc/training/scientific/short_courses/Shell_Scripting.ppt



Hands-on Exercises

1. The simplest Hello World shell script - Echo command
2. Summation of two integers - If block
3. Summation of two real numbers - bc (basic calculator) command
4. Script to find out the biggest number in 3 numbers - If -elif block
5. Operation (summation, subtraction, multiplication and division) of two numbers - Switch
6. Script to reverse a given number - While block
7. A more complicated greeting shell script
8. Sort the given five numbers in ascending order (using array) - Do loop and array
9. Calculating average of given numbers on command line arguments - Do loop
10. Calculating factorial of a given number - While block
11. An application in research computing - Combining all above
12. **Optional:** Write own shell scripts for your own purposes if time permits

The PPT/WORD format of this presentation is available here:

<http://its2.unc.edu/divisions/rc/training/scientific/>

/afs/isis/depts/its/public_html/divisions/rc/training/scientific/short_courses/ 72

Presentation Topic

Cloud Computing

Vishal Ambadas Meshram
vishal.meshram@viit.ac.in

Department of Computer Engineering



BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

UNIT 1:

Introduction To Cloud Computing

Reference:

- 1) Thomas Erl, Zaigham Mahmood and Ricardo Puttini, “*Cloud Computing: Concepts, Technology and Architecture*”,
- 2) Wikipedia
- 3) Edureka

Contents

- Overview, Applications, Intranets and the Cloud.
- Your Organization and Cloud Computing- Benefits, Limitations, Security Concerns.
- Software as a Service (SaaS)- Understanding the Multitenant Nature of SaaS Solutions,
- Understanding SOA.
- Platform as a Service (PaaS).
- Infrastructure as a Service (IaaS)
- Case Study: Google Cloud Platform

CO Achieved in First Lecture

- 1) To understand cloud computing concepts
- 2) To study supporting technologies of cloud

Course Outcomes:-

At the end of the unit you will be able to:

- 1) Summarize the basic concepts of cloud computing
(Understand)
- 2) Make use of supporting technologies for cloud computing
(Understand, Apply)

University Questions

Q1)

- a) Comment on “Cloud and virtualization”
- b) What do you mean by pods, aggregation and silos in terms of IAAS
- c) Enlist and explain the challenges and obstacles while adopting cloud computing?

OR

Q2)

- a) What is administrating and monitoring cloud services? Enlist and explain different tools used for administrating and monitoring cloud services.
- b) Enlist services provided by SAAS, PAAS and IAAS
- c) How do you relate cloud computing with utility computing?

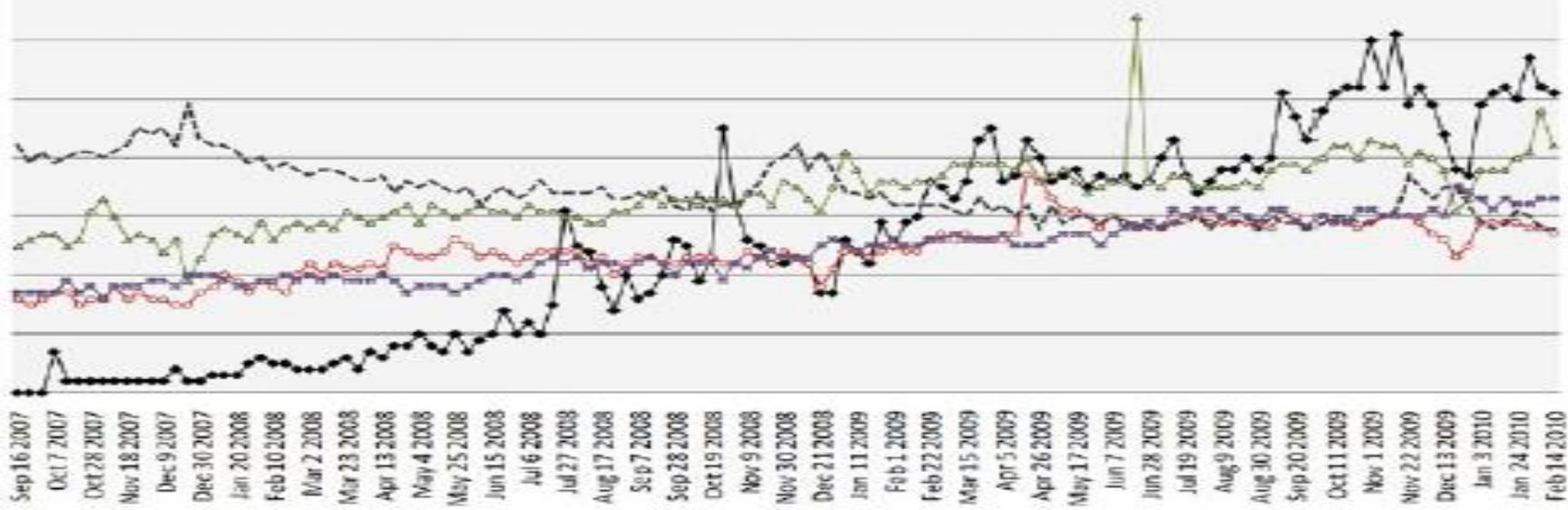
Popularity of cloud computing

Top 10 Strategic Technology Areas for 2010 (Powered By Gartner)

- 1. Cloud Computing
- 2. Advanced Analytics
- 3. Client Computing
- 4. IT for Green
- 5. Reshaping the Data Center
- 6. Social Computing
- 7. Security – Activity Monitoring
- 8. Flash Memory
- 9. Virtualization for Availability
- 10. Mobile Applications

Top Strategic Technologies (Powered By Google Trends)

—●— Cloud Computing —●— GreenIT —●— Social Network —●— Flash Memory —●— Mobile Application



A brief history

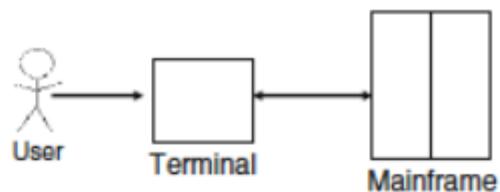
- The idea of computing in a “cloud” traces back to the origins of utility computing, a concept that computer scientists **John McCarthy** publically proposed in **1961**:

“ If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility..... the computer utility could become the basis of a new and important industry”

History

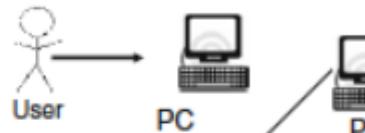
Many users shared powerful mainframes using dummy terminals

1. Mainframe Computing



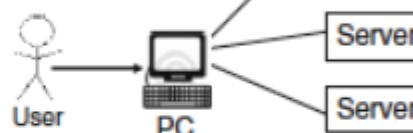
Stand-alone PCs became powerful enough to meet the majority of users' needs

2. PC Computing



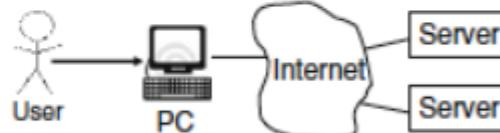
PCs, laptops, and servers were connected together through local networks to share resources and increase performance

3. Network Computing



Local networks were connected to other local networks forming a global network such as the Internet to utilise remote applications and resources

4. Internet Computing



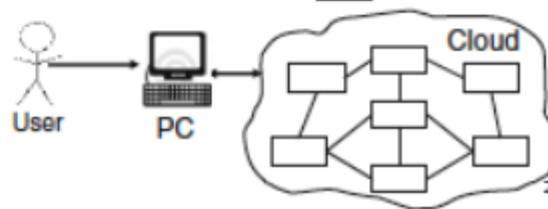
Grid computing provided shared computing power and storage through a distributed computing system

5. Grid Computing



Cloud computing further provides shared resources on the Internet in a scalable and simple way

6. Cloud Computing



Borko Furht, "Handbook of Cloud Computing"

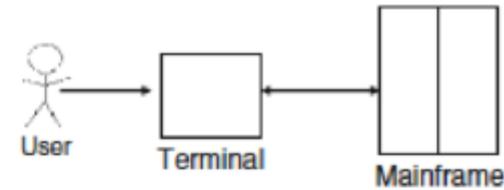
29

History

Offers finite computing power

Dummy terminals acted as user interface devices

1. Mainframe Computing

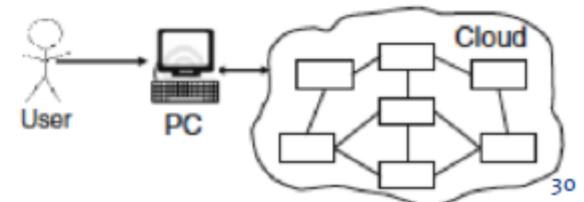


Quite similar?

Provides almost infinite power and capacity

Powerful PCs can provide local computing power and caching support

6. Cloud Computing



Borko Furht, "Handbook of Cloud Computing"

30

Introduction to Cloud Computing

- **Cloud Computing can be defined as**

“a new style of computing in which dynamically scalable and often virtualized resources are provided as a services over the Internet”.

Forrester Research provided its own definition of cloud computing:

“.... A standardized IT capability (services, software, or infrastructure) delivered via internet technologies in a pay-per-use, self-service way.”

Introduction to Cloud Computing

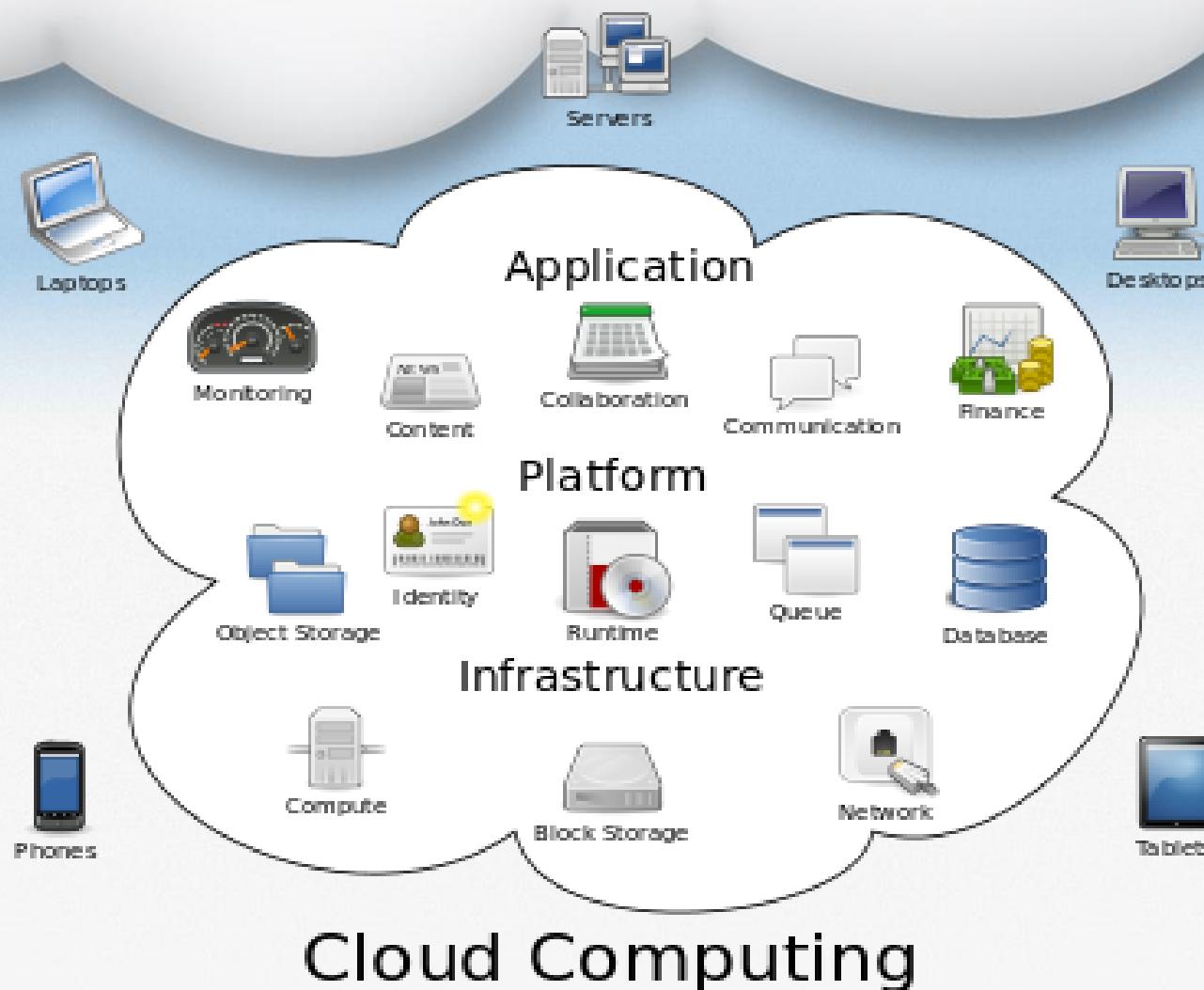
- **NIST (National Institute of Standards & Technology) Definition:**

*“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. *network, servers, storage, applications, and services*) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. ”*

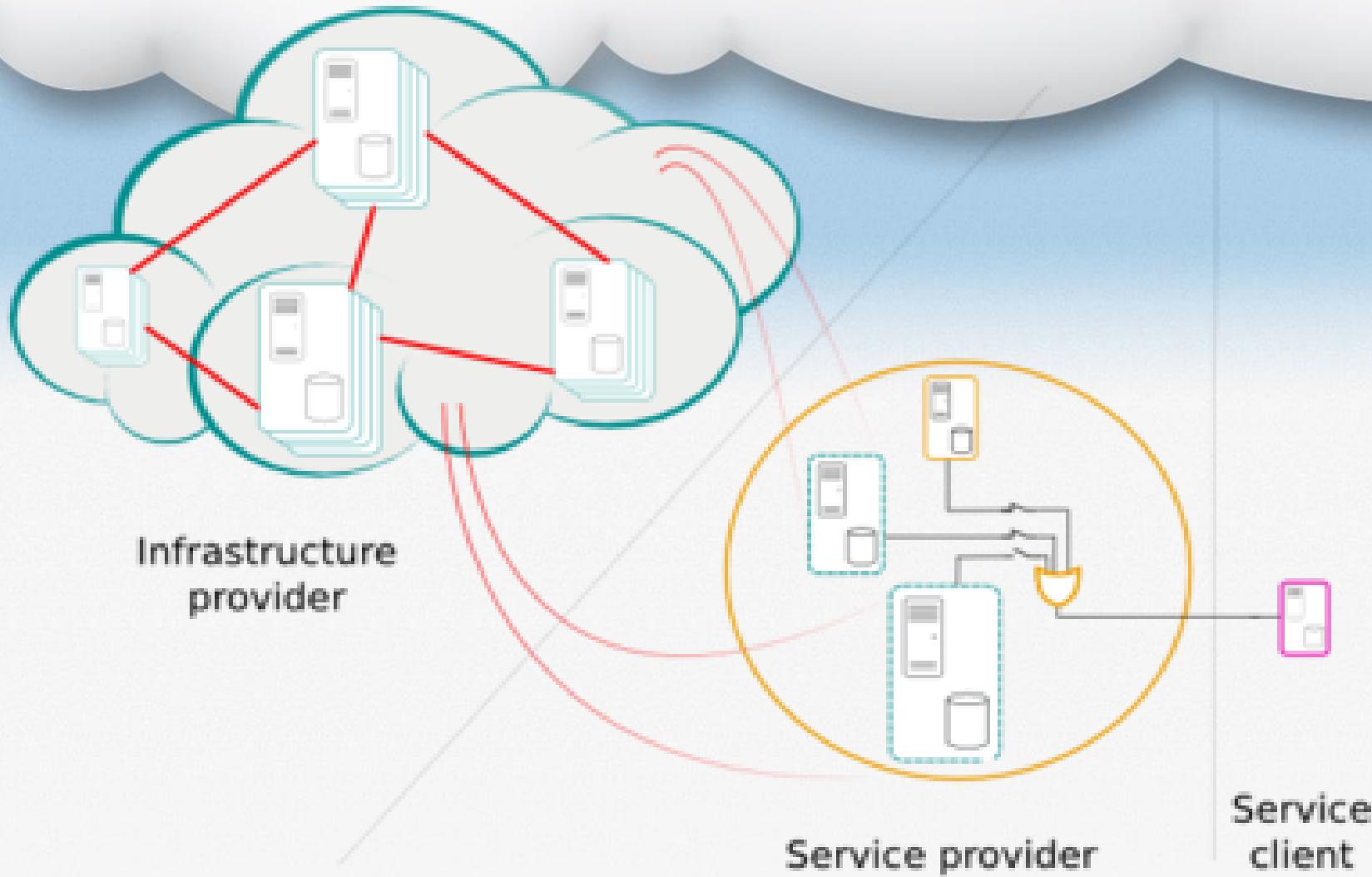
- A more concise definition:

“Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources”

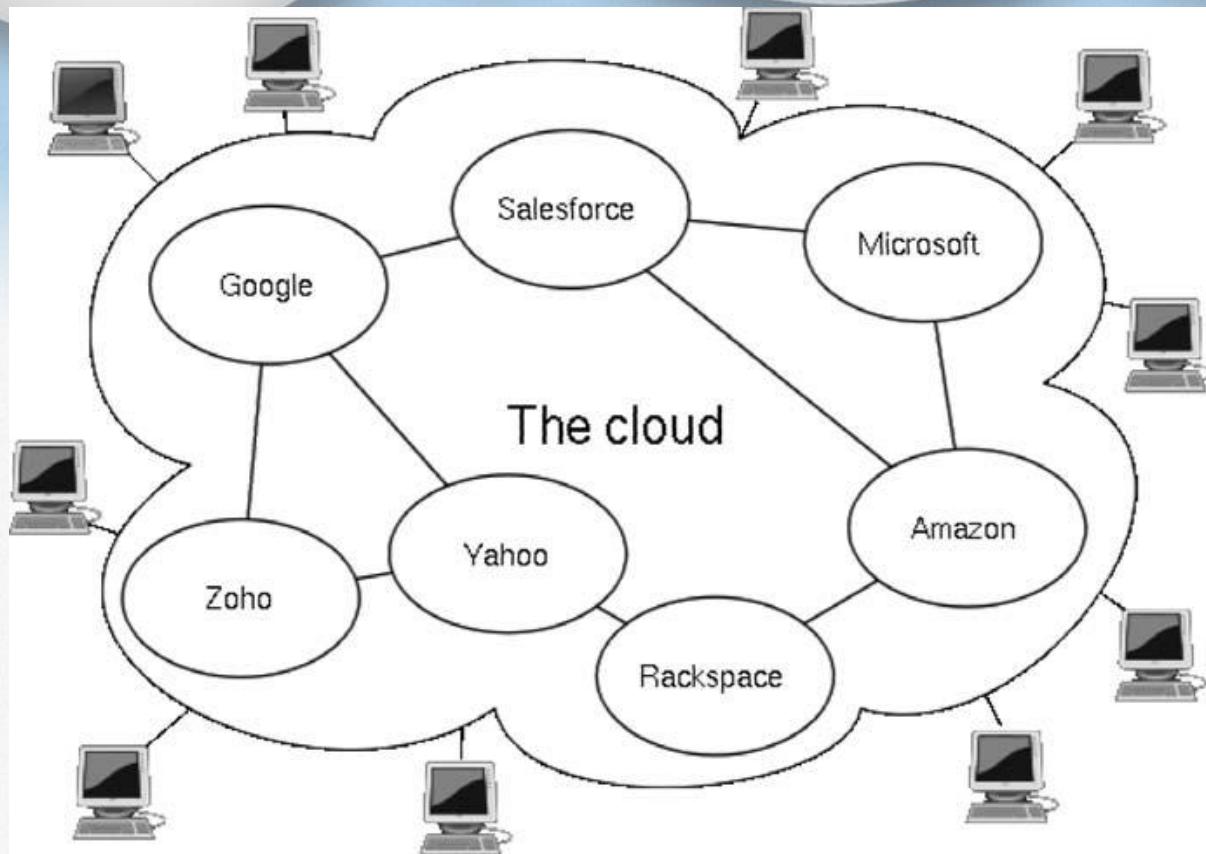
What is cloud computing?



What is cloud computing?



What is cloud computing?



Business Drivers

- Capacity Planning
- Cost Reduction
- Organizational Agility

Technology Innovations

- Clustering:

A cluster is a group of independent IT resources that are interconnected and work as a single system. System failure rates are reduced while availability and reliability are increased, since redundancy and failover are inherent to the cluster.

A general prerequisite of hardware clustering is that its component systems have reasonably identical hardware and operating systems to provide similar performance levels when one failed component is to be replaced by another. Component devices that form a cluster are kept in synchronization through dedication, high-speed communication links.

the basic concept of built-in redundancy and failover is core to cloud platforms.

Technology Innovations

- Grid Computing:

A computing grid (or “Computational grid”) provides a platform in which computing resources are organized into one or more logical pools. These pools are collectively coordinated to provide a high performance distributed grid, sometimes referred to as a “super virtual computer”.

Grid computing differs from clustering in that grid systems are much loosely coupled and distributed. As a result, grid computing systems can involve computing resources that are heterogeneous and geographically dispersed, which is generally not possible with cluster-based systems.

The technological advancements achieved by grid computing projects have influenced various aspects of cloud computing platforms and mechanism, specifically in relation to common feature- sets such as networked access, resource pooling, and scalability and resiliency.

These types of features can be established by both grid computing and cloud computing, in their own distinctive approaches.

Technology Innovations

- Virtualization:

Virtualization represents a technology platform used for the creation of virtual instances of IT resources.

A layer of virtualization software allows physical IT resources to provide multiple virtual images of themselves so that their underlying processing capabilities can be shared by multiple users.

Prior to the virtualization technology, software was limited to residing on and being coupled with static hardware environments. The virtualization process servers this software-hardware dependency, as hardware requirements can be simulated by emulation software running in virtualized environments.

Established virtualization technologies can be traced to several cloud characteristics and cloud computing mechanisms, having inspired many of their core features.

Modern virtualization technologies emerged to overcome the performance, reliability and scalability limitations of traditional virtualization platforms.

Technology Innovations Vs. Enabling Technologies

- It is essential to highlight several other areas of technology that continue to contribute to modern-day cloud-based platforms. There are distinguished as cloud-enabling technologies :
 - Broadband Networks and internet architecture
 - data center technology
 - (Modern) virtualization technology
 - Web technology
 - Multitenant technology
 - Service technology

Each of these cloud-enabling technologies existed in some form prior to the formal advent of cloud computing some were refined further, and on occasion even redefined, as a result of the subsequent evolution of cloud computing.

Summary of key points

- The primary business drivers that exposed the need of cloud computing and led to its formation include capacity planning, cost reduction, and organizational agility.
- The primary technology innovations that influenced and inspired key distinguishing features and aspects of cloud computing include clustering, grid computing, and traditional forms of virtualization.

Home Assignment Question

- Difference between Cluster, Grid and cloud computing??
- Tomorrow's Points:
 - Five essential characteristics,
 - Three service models, and
 - Four deployment models.

Muddiest Point

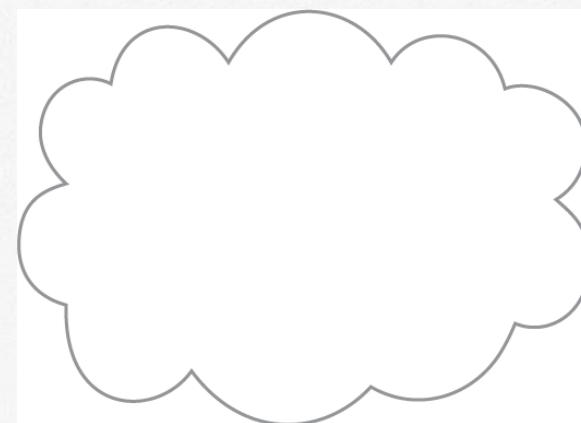
Thank You!!!!

Basic Concepts & Terminology

Cloud:

- A cloud refers to a distinct IT environment that is designed for the purpose of remotely provisioning scalable and measured IT resources.
- the symbol of cloud was commonly used to represent the Internet in a verity of specifications.
- This same symbol is now used to specifically represent the boundary of cloud environment, as shown in fig.

Fig: The symbol used to denote the boundary of a cloud environment.



Copyright © Arcitura Education

Basic Concepts & Terminology

Difference between “Cloud” and Internet :

1. As a specific environment used to remotely provision IT resources, a cloud has a finite boundary.
whereas the Internet provides open access to many Web-based IT resources, a cloud is typically privately owned and offers access to IT resources that is metered.

2. Much of the internet is dedicated to the access of content-based IT resources published via the World Wide Web,
on the other hand, are dedicated to supplying back-end processing capabilities and user-based access to these capabilities.

Basic Concepts & Terminology

□ IT Resources:

- An IT resource is a physical or virtual IT-related artifact that can be either software-based, such as virtual server or a custom software program, or hardware-based, such as a physical server or a network device.

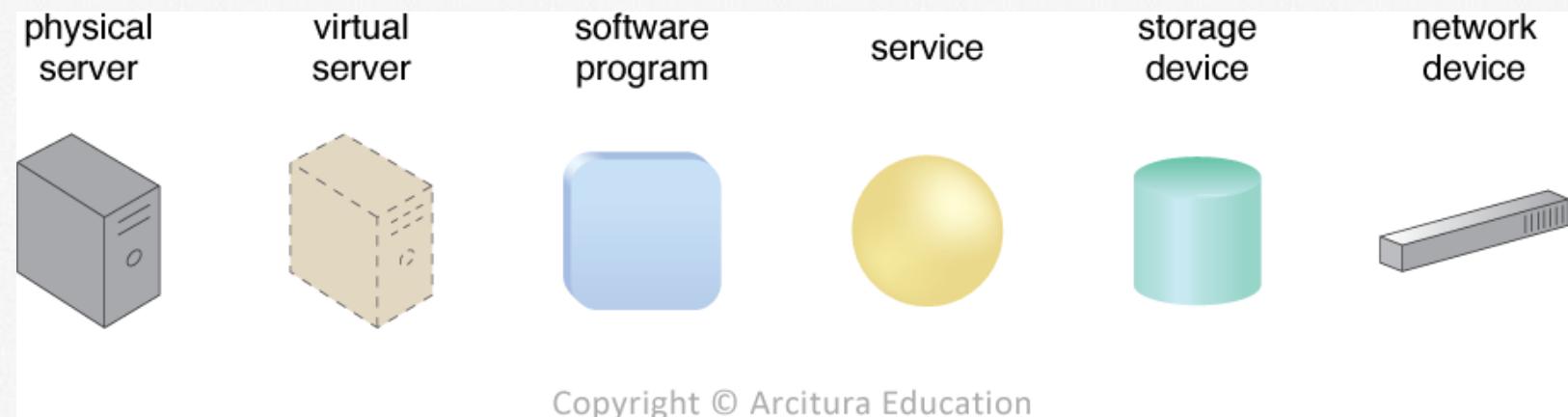


Figure: Examples of common IT resources and their corresponding symbols

Basic Concepts & Terminology

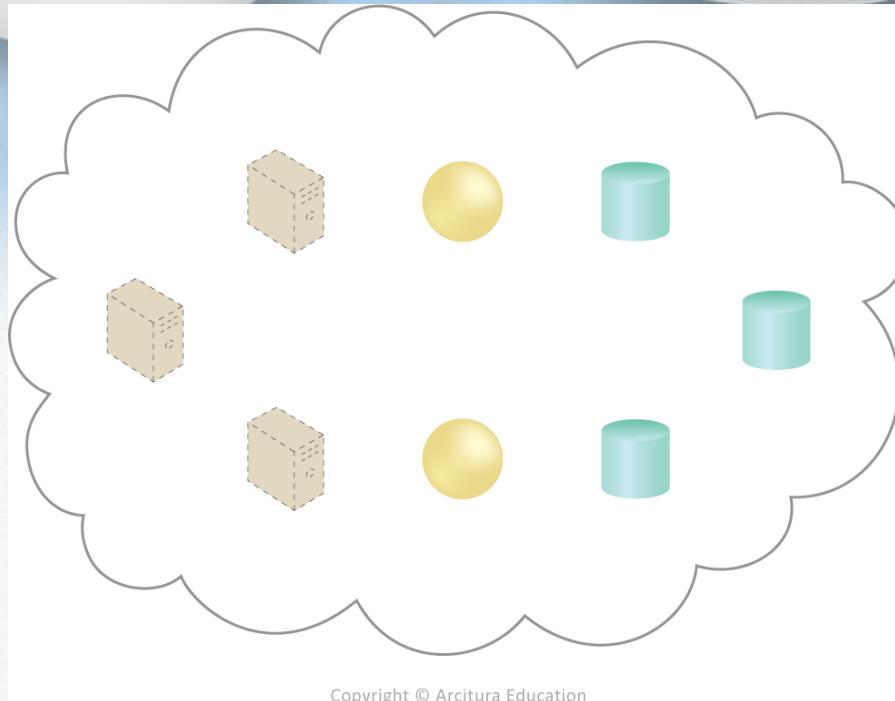


Figure: A cloud is hosting eight IT resources: three virtual servers, two cloud services, and three storage devices.

Basic Concepts & Terminology

On-Premise:

- As a distinct and remotely accessible environment, a cloud represents an option for the deployment of IT resources.
- An IT resources that hosted in a conventional IT enterprise within an organizational boundary (that does not specifically represent a cloud) is considered to be located on the premises of the IT enterprise, or on-premise for short.
- ***“on the premises of a controlled IT environment that is not cloud-based”***
- ***An IT resources that is on-premise cannot be a cloud-based, and vice-versa.***

Basic Concepts & Terminology

Scaling:

Scaling, from an IT recourse perspective, represents the ability of the IT resource to handle increased or decreased usage demands.

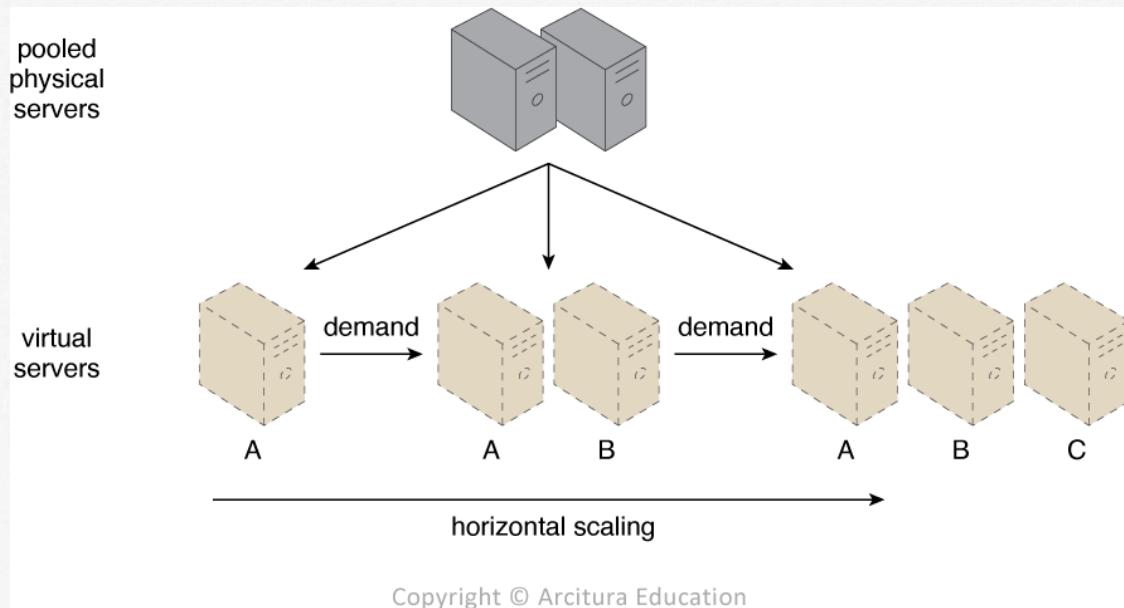
The following are types of scaling:

- Horizontal Scaling : scaling out and scaling in
- Vertical Scaling : scaling up and scaling down

Basic Concepts & Terminology

□ Horizontal Scaling :

- ✓ The allocating or releasing of IT resources that are of the same type is referred to as **horizontal scaling**, as shown in fig.
- ✓ The horizontal allocation of resources is referred to as **scaling out** and the horizontal releasing of resources is referred to as **scaling in**.
- ✓ Horizontal scaling is a common form of scaling within cloud environment.

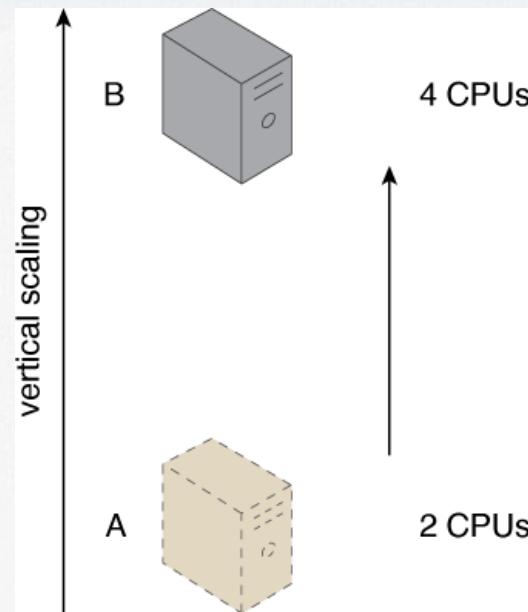


Basic Concepts & Terminology

□ Vertical Scaling :

- ✓ When an existing IT resources is replaced by another with higher or lower capacity, vertical scaling is considered to have occurred, as shown in the fig.
- ✓ The replacing of an IT resource with higher capacity is referred to as scaling up and the replacing an IT resource with lower capacity is referred to as scaling down.

Figure: An IT resource (a virtual server with two CPUs) is scaled up by replacing it with a more powerful IT resource with increased capacity for data storage (a physical server with four CPUs)

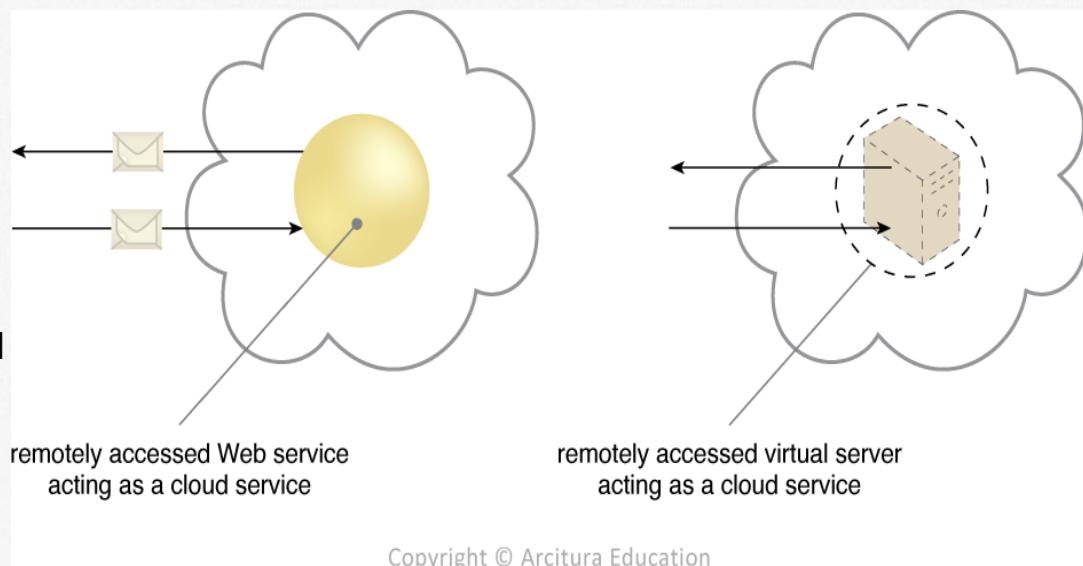


Basic Concepts & Terminology

□ Cloud Service:

- A **cloud service** is any IT resource that is made remotely accessible via a cloud.
- The term “service” within the context of cloud computing is especially broad.
- A cloud service can exist as a simple web-based software program with a technical interface invoked via the use of a messaging protocol, or as a remote access point for administrative tools or large environment and other IT resources.

Figure: A cloud service with a published technical interface is being accessed by a consumer outside of the cloud (left). A cloud service that exists as a virtual server is also being accessed from outside of the cloud's boundary (right)



Copyright © Arcitura Education

Basic Concepts & Terminology

□ Cloud Service consumers:

- A **cloud service consumer** is a temporary runtime role assumed by a software program when it accesses a cloud service.
- As shown in the fig. common type of cloud service consumers can include software programs and services capable of remotely accessing cloud services with published service contracts.



Figure: Examples of cloud service consumers. Depending on the nature of a given diagram, an artifact labeled as a cloud service consumer may be a software program or a hardware device (in which case it is running a software program capable of acting as a cloud service consumer).

Goals & Benefits

- Reduced Investments and proportional costs
- Increased scalability
- Increased availability and reliability

Cloud Reference Model

- The design of the NIST cloud computing reference architecture serves the following objectives:
 - to illustrate and understand the various cloud services in the context of an overall cloud computing conceptual model;
 - to provide a technical reference to USG agencies and other consumers to understand, discuss, categorize and compare cloud services;
 - and to facilitate the analysis of candidate standards for security, interoperability, and portability and reference implementations.

Cloud Reference Model

3. Cloud Computing Reference Architecture

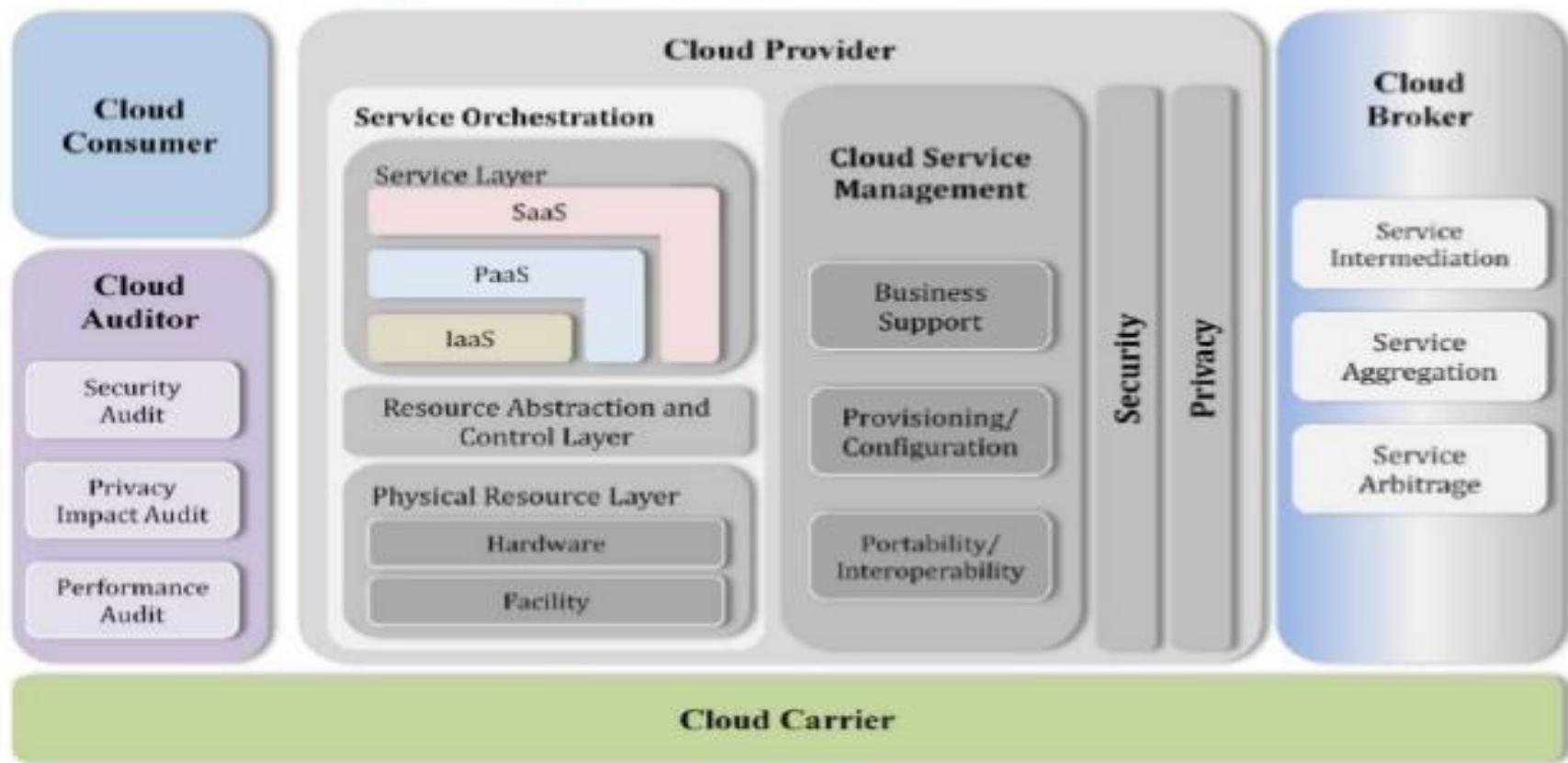


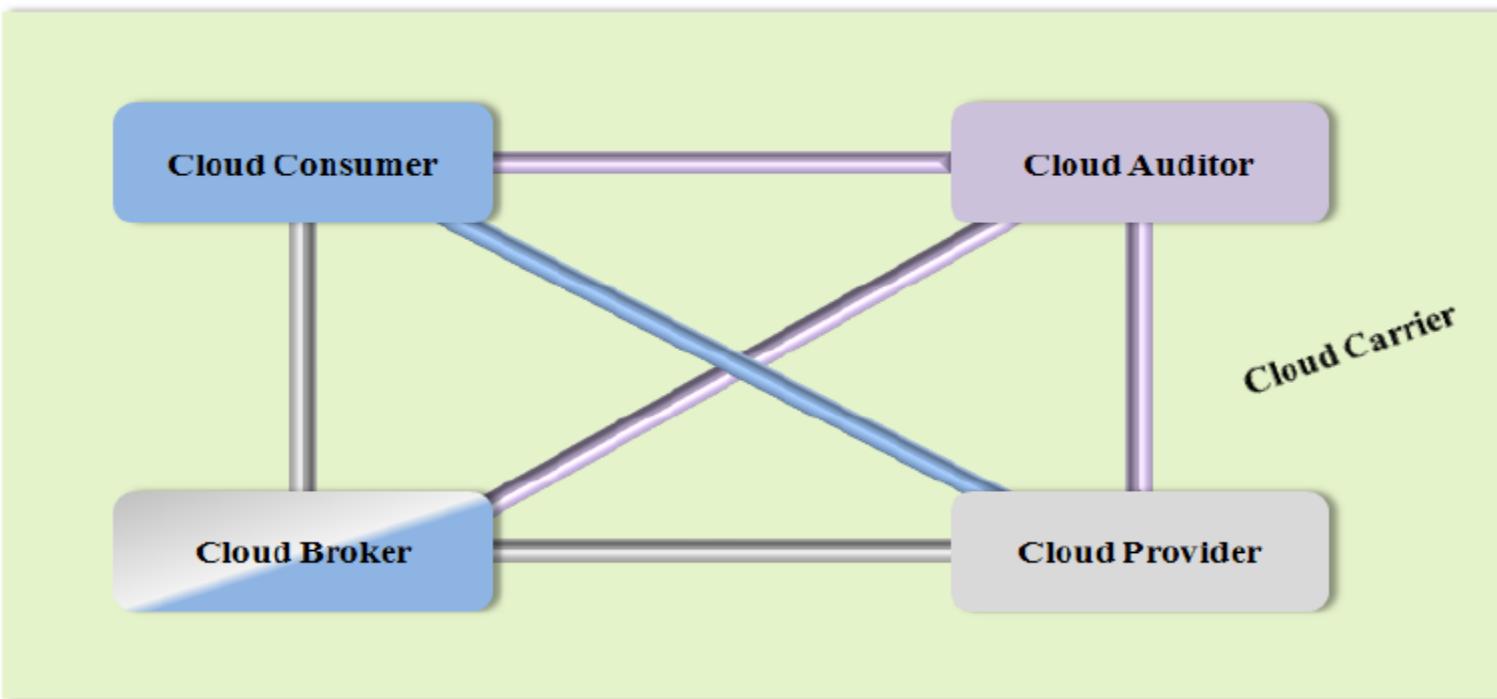
Figure 1: The Conceptual Reference Model

Roles in Cloud Computing

Actor	Definition
Cloud Consumer	A person or organization that maintains a business relationship with, and uses service from, <i>Cloud Providers</i> .
Cloud Provider	A person, organization, or entity responsible for making a service available to interested parties.
Cloud Auditor	A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between <i>Cloud Providers</i> and <i>Cloud Consumers</i> .
Cloud Carrier	An intermediary that provides connectivity and transport of cloud services from <i>Cloud Providers</i> to <i>Cloud Consumers</i> .

Table 1: Actors in Cloud Computing

Communication between actors



- The communication path between a cloud provider and a cloud consumer
- The communication paths for a cloud auditor to collect auditing information
- The communication paths for a cloud broker to provide service to a cloud consumer

Figure 2: Interactions between the Actors in Cloud Computing

Cloud Usage Scenario Examples

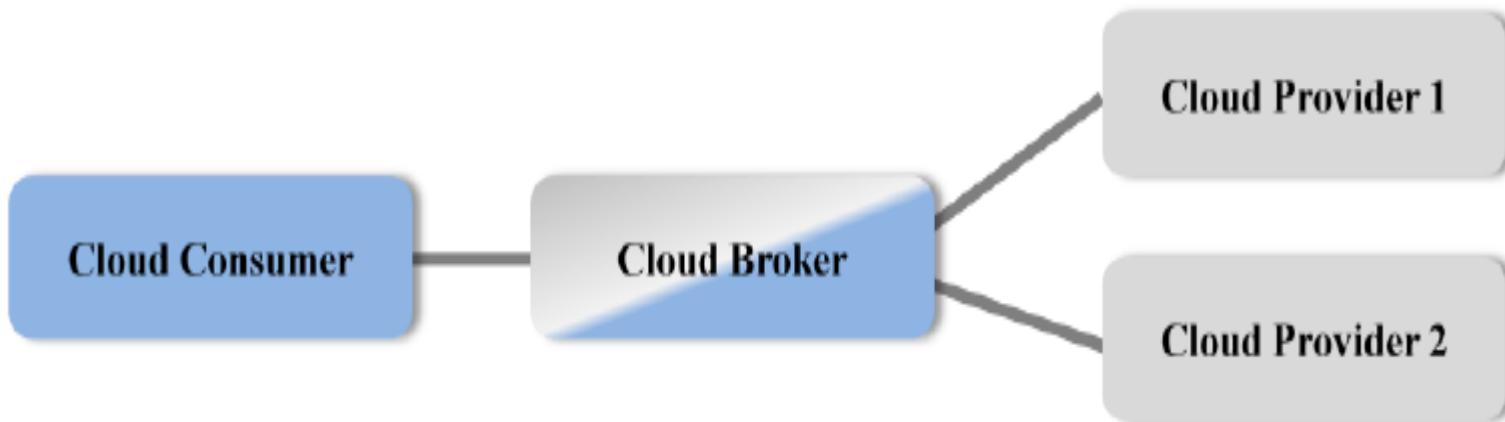
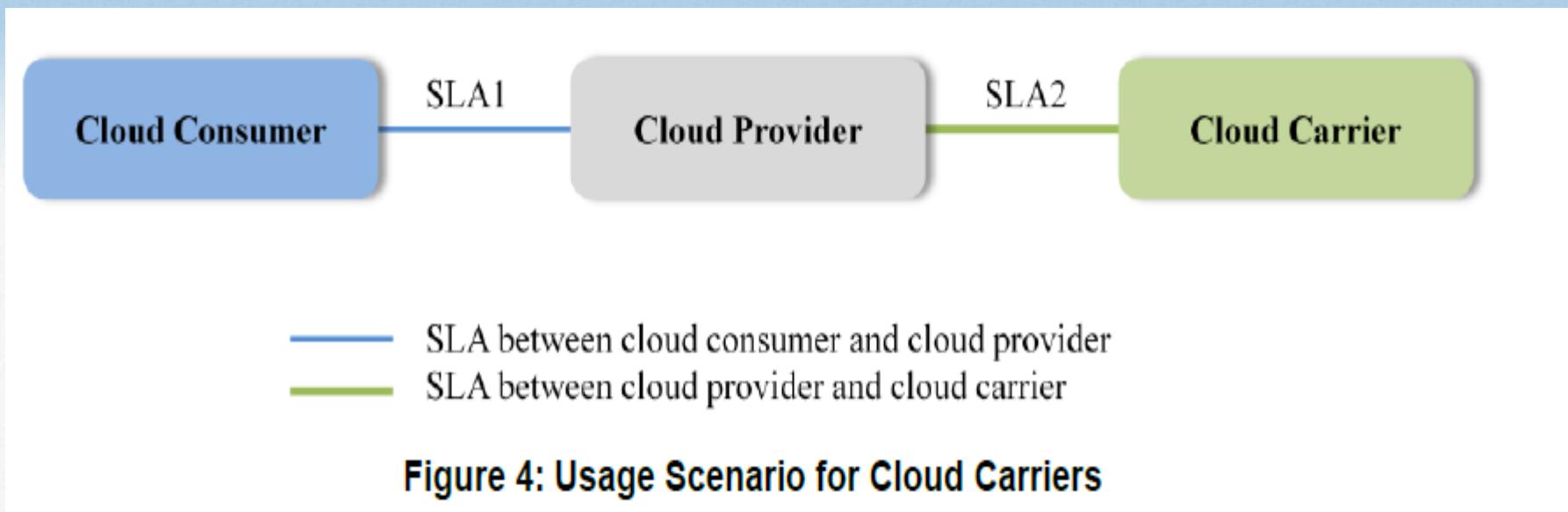


Figure 3: Usage Scenario for Cloud Brokers

Cloud Usage Scenario Examples



Cloud Usage Scenario Examples

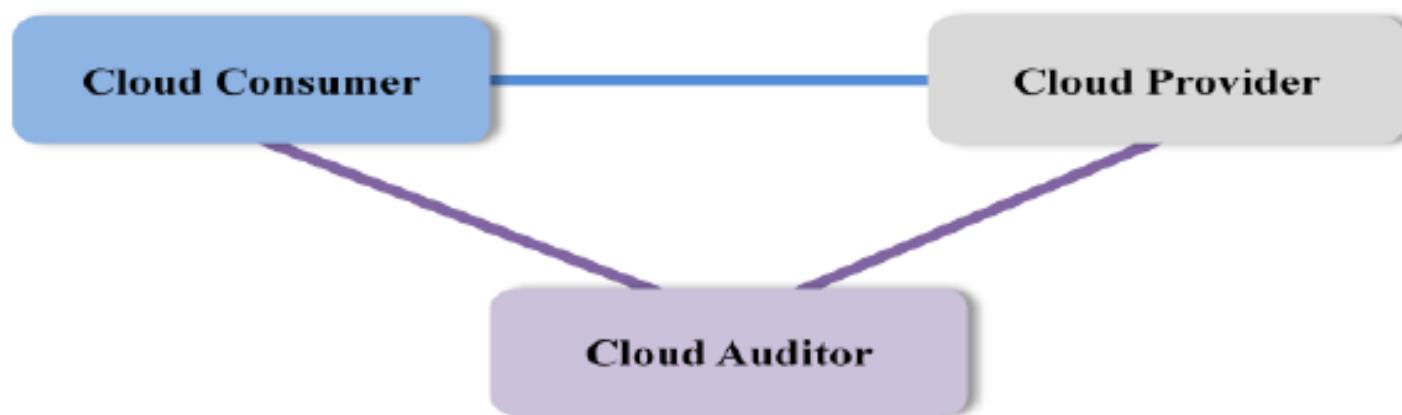
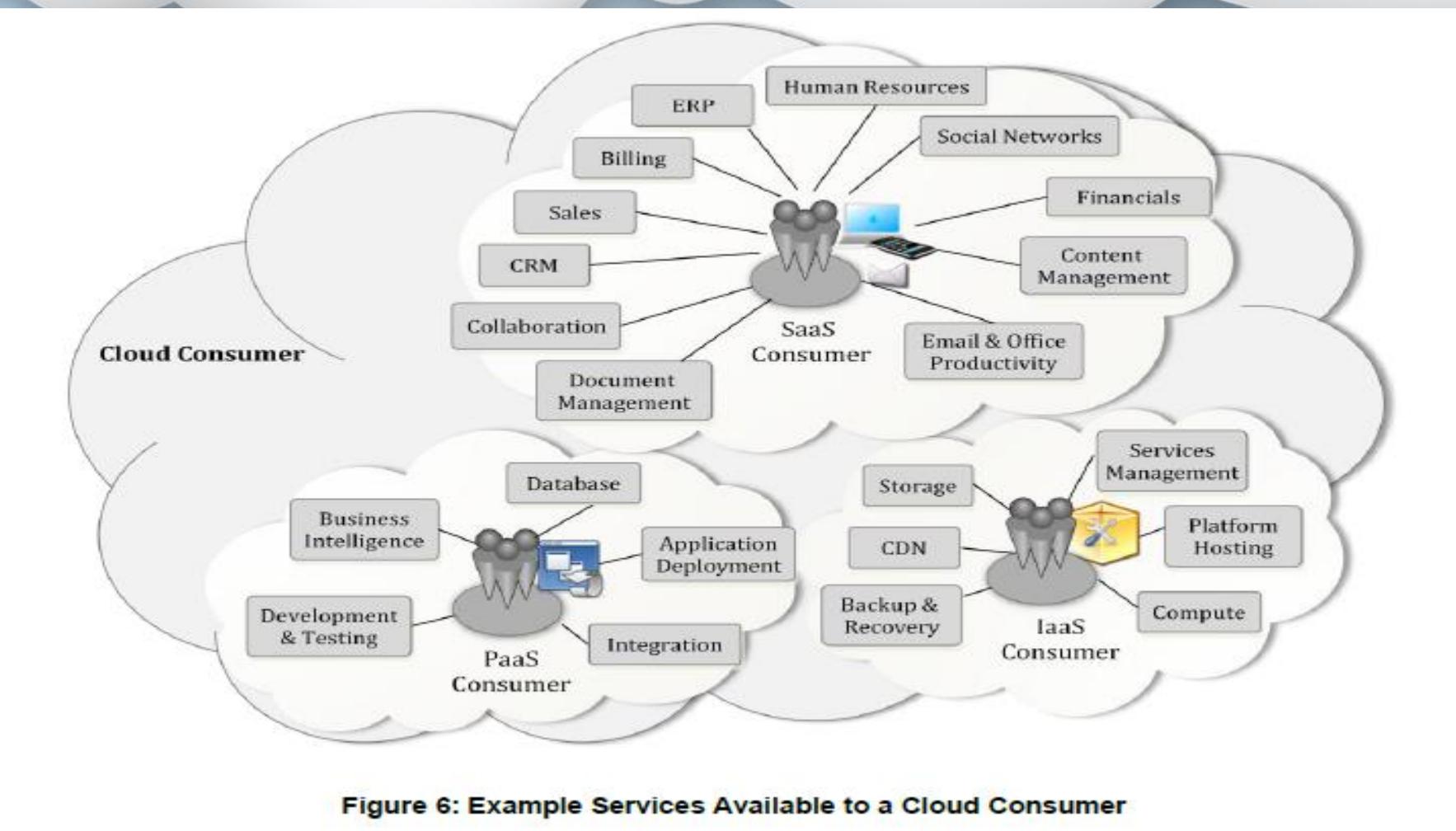


Figure 5: Usage Scenario for Cloud Auditors

Cloud Consumer



Cloud Service Provider

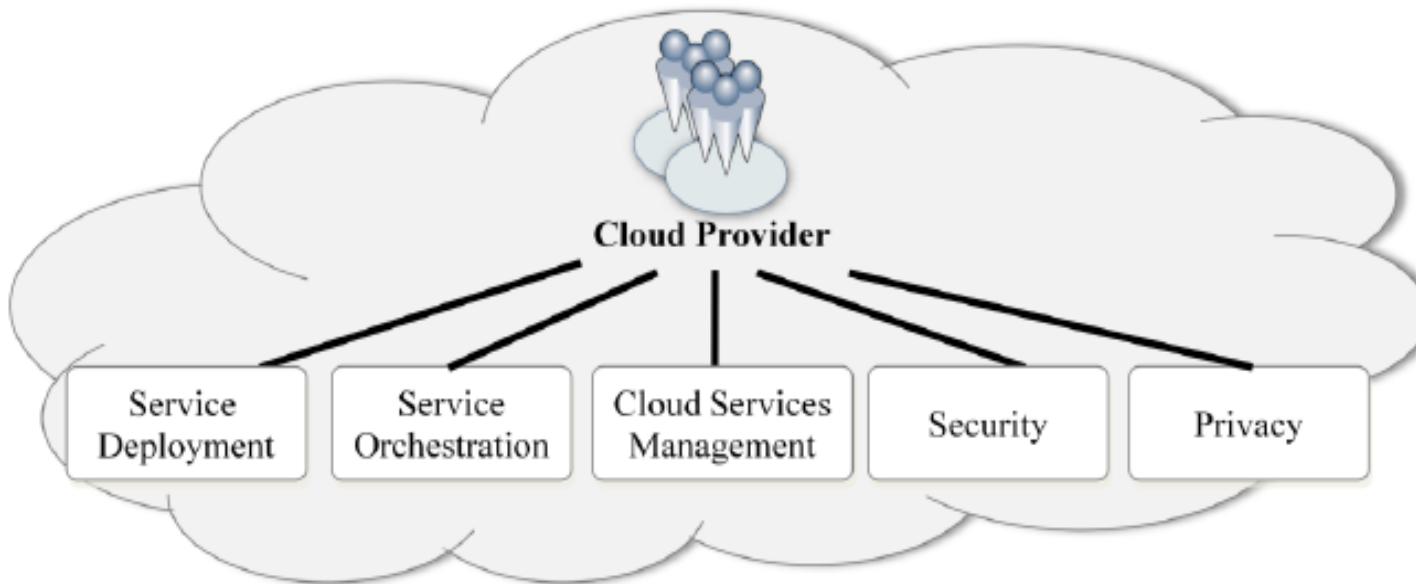


Figure 7: Cloud Provider - Major Activities

Cloud Auditor

- A cloud auditor is a party that can perform an independent examination of cloud service controls with the intent to express an opinion thereon.
- Audits are performed to verify conformance to standards through review of objective evidence.
- A cloud auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, performance, etc.
- **COMPANY EXAMPLES:**
 - ISACA (Infor. System audit & Control Association)
 - Product of ISACA is COBIT 5. (Control Objectives for Information and Related Technologies)
 - CloudChecker.com
 - Etc....

Cloud Broker

- As cloud computing evolves, the integration of cloud services can be too complex for cloud consumers to manage.
- A cloud consumer may request cloud services from a cloud broker, instead of contacting a cloud provider directly.
- A cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.
- Cloud Broker Companies: (*Total market is of \$2.03 billions in 2018*)
 - Appirio
 - AWS Marketplace
 - Bluewolf
 - CloudCompare
 - RED HAT Cloudforms is multicloud management platform.

Cloud Broker

In general, a cloud broker can provide services in three categories:

- **Service Intermediation:** A cloud broker enhances a given service by improving some specific capability and providing value-added services to cloud consumers. The improvement can be managing access to cloud services, identity management, performance reporting, enhanced security, etc.
- **Service Aggregation:** A cloud broker combines and integrates multiple services into one or more new services. The broker provides data integration and ensures the secure data movement between the cloud consumer and multiple cloud providers.
- **Service Arbitrage:** Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The cloud broker, for example, can use a credit-scoring service to measure and select an agency with the best score.

Cloud Carrier

- A cloud carrier acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers.
- Cloud carriers provide access to consumers through network, telecommunication and other access devices. For example, cloud consumers can obtain cloud services through network access devices, such as computers, laptops, mobile phones, mobile Internet devices (MIDs), etc.
- The distribution of cloud services is normally provided by network and telecommunication carriers or a *transport agent*, where a transport agent refers to a business organization that provides physical transport of storage media such as high-capacity hard drives.

Scope of Control between Provider & Consumer

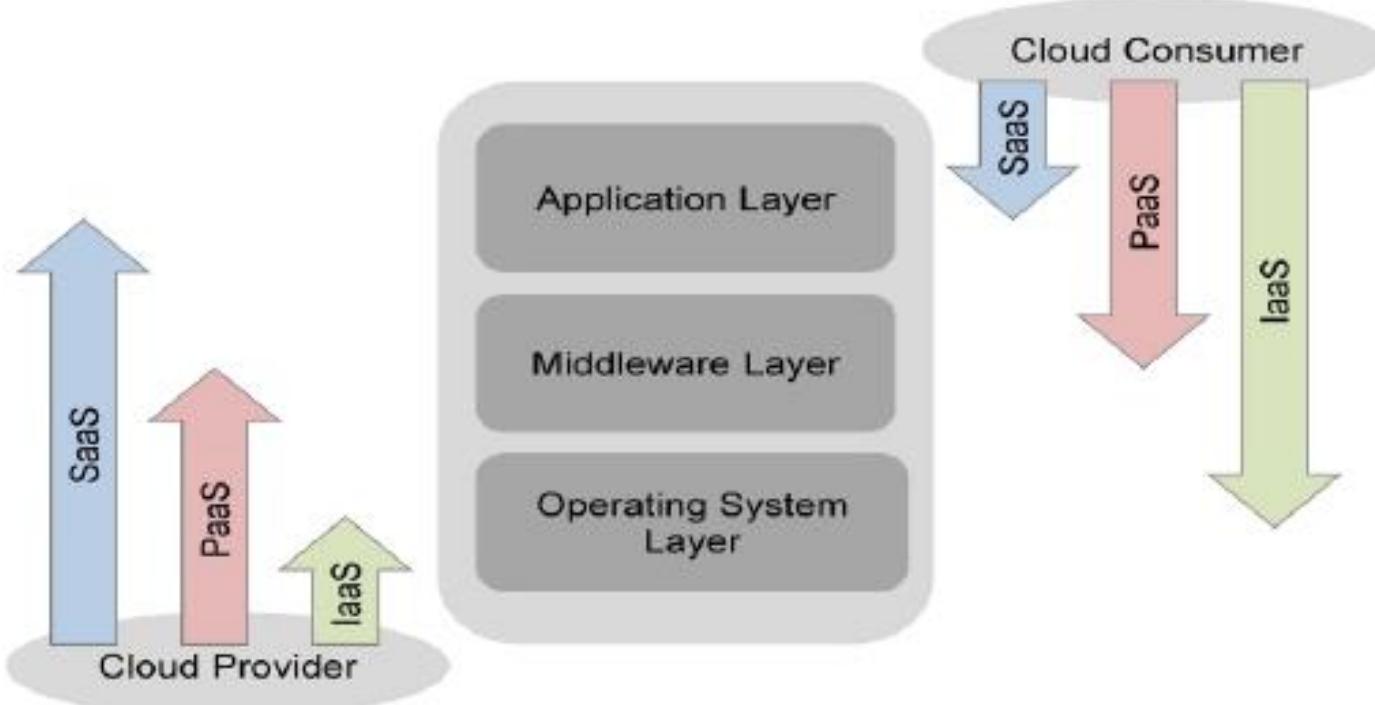
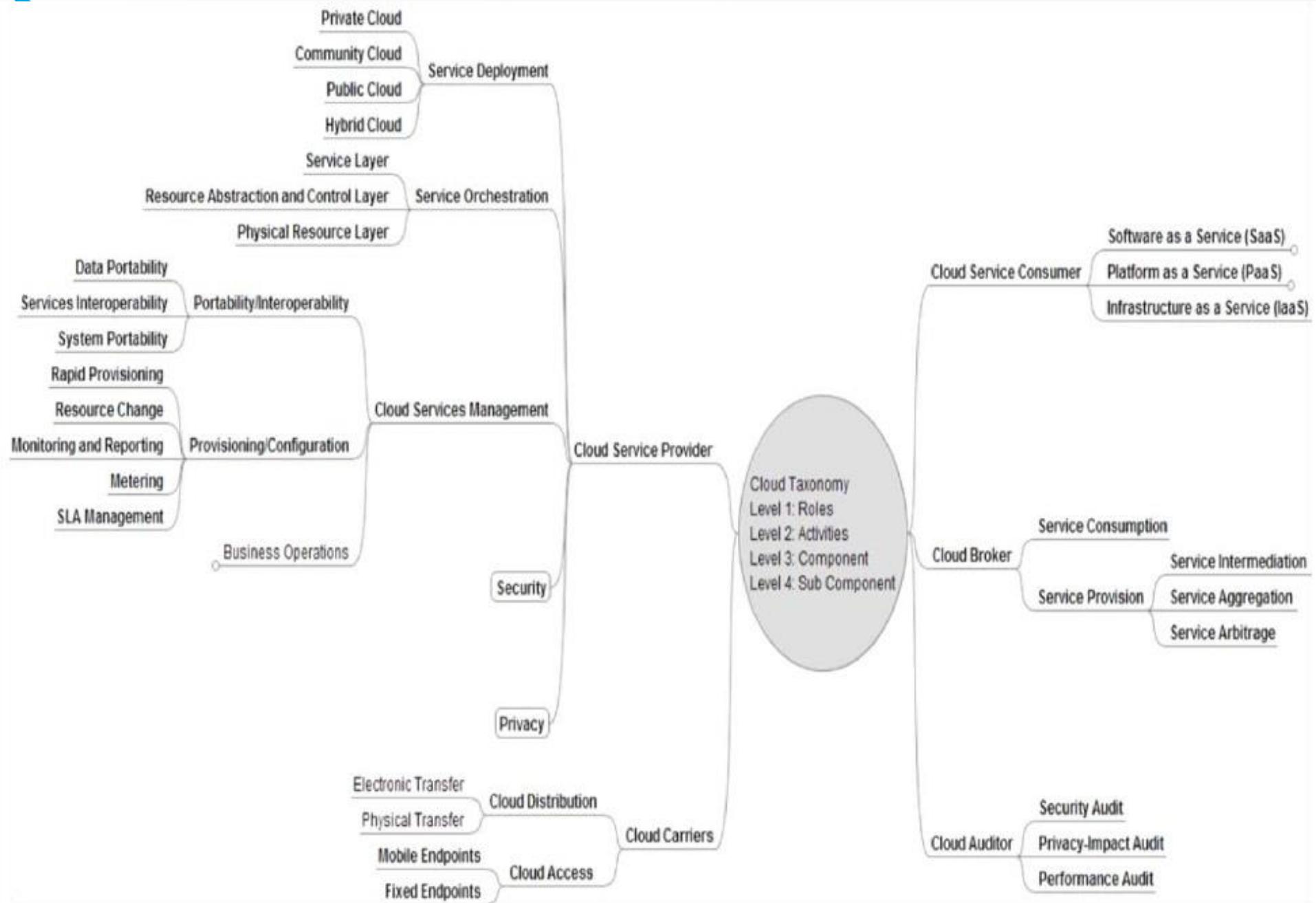


Figure 8: Scope of Controls between Provider and Consumer

Cloud Taxonomy

- Taxonomy is the science of categorization, or classification, of things based on a predefined system.
- Typically, taxonomy contains a controlled vocabulary with a hierarchical tree-like structure.



Cloud Taxonomy

- Figure presents the taxonomy associated with the cloud computing reference architecture discussed in this document. In the figure, a four-level taxonomy is presented to describe the key concepts about cloud computing.
- **Level 1:** *Role*, which indicates a set of obligations and behaviors as conceptualized by the associated actors in the context of cloud computing.
- **Level 2:** *Activity*, which entails the general behaviors or tasks associated to a specific role.
- **Level 3:** *Component*, which refer to the specific processes, actions, or tasks that must be performed to meet the objective of a specific activity.
- **Level 4:** *Sub-component*, which present a modular part of a component.

Roles

- **Cloud Consumer** - Person or organization that maintains a business relationship with, and uses service from, Cloud Service Providers.
-
- **Cloud Provider** – Person, organization or entity responsible for making a service available to service consumers.
-
- **Cloud Carrier** – The intermediary that provides connectivity and transport of cloud services between Cloud Providers and Cloud Consumers.
-
- **Cloud Broker** – An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers.
-
- **Cloud Auditor** – A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation.

Cloud Service provider activities

1. **Service Deployment** – All of the activities and organization needed to make a cloud service available
2. **Service Orchestration** - Refers to the arrangement, coordination and management of cloud infrastructure to provide different cloud services to meet IT and business requirements.
3. **Cloud Service Management** – Cloud Service Management includes all the service-related functions that are necessary for the management and operations of those services required by or proposed to customers.

Cloud Service provider activities

4. **Security** – Refers to information security. “information security” means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide:
 - (A) **integrity**, which means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity;
 - (B) **confidentiality**, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information;
 - (C) **availability**, which means ensuring timely and reliable access to and use of information.
5. **Privacy** - Information privacy is the assured, proper, and consistent collection, processing, communication, use and disposition of disposition of personal information (PI) and personally-identifiable information (PII) throughout its life cycle.

Cloud Carrier activities:

1. **Cloud Distribution** – The process of transporting cloud data between Cloud Providers and Cloud Consumers.
2. **Cloud Access** – To make contact with or gain access to Cloud Services.

Risks & Challenges

- Increased security vulnerabilities.

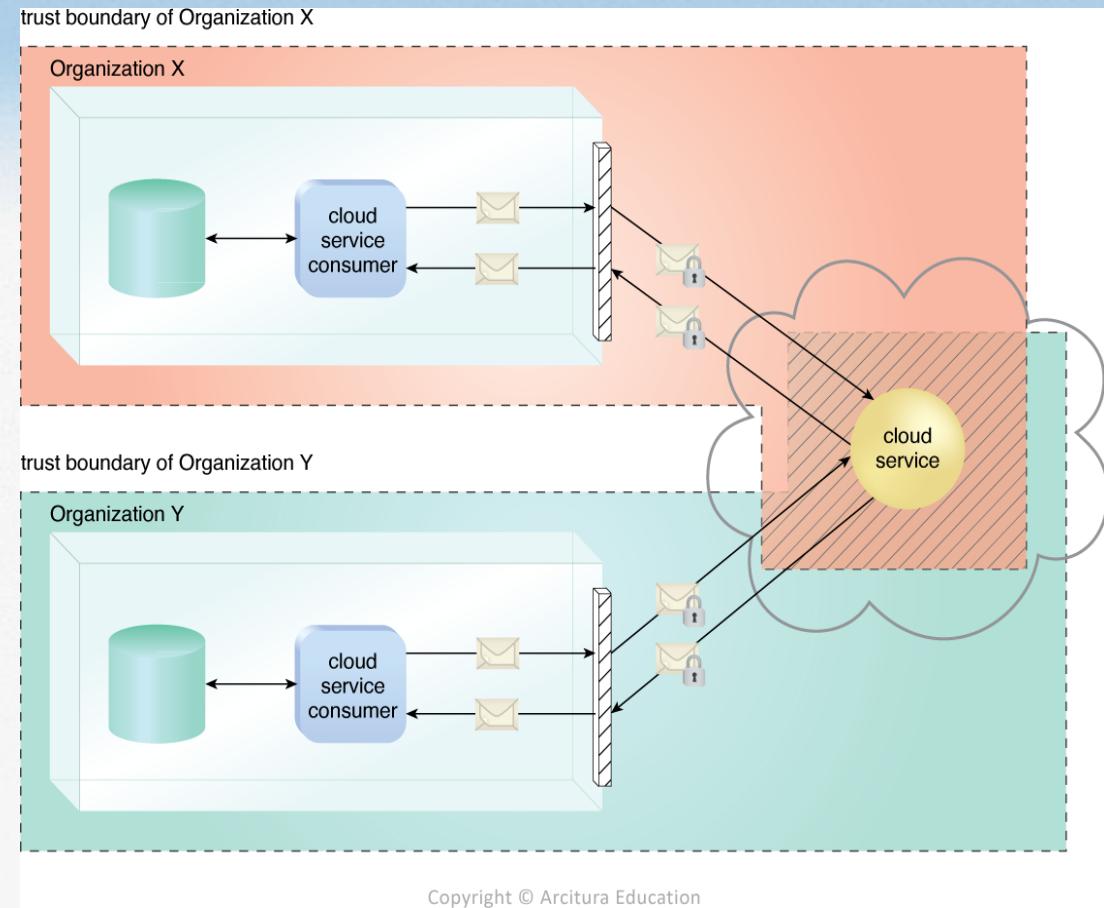
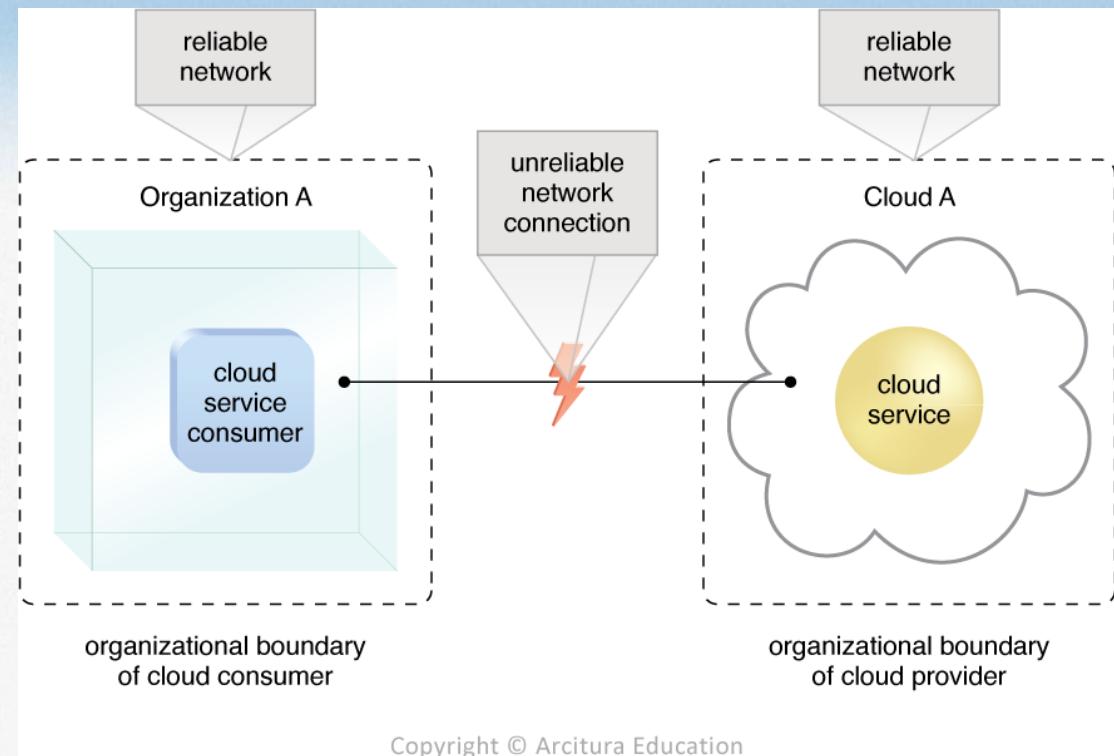


Figure: The shaded area with diagonal lines indicates the overlap of two organization's trust boundaries.

Risks & Challenges

- Reduced Operational Governance Control

Figure: An unreliable network connection compromises the quality of communication between cloud consumer and cloud provider environments.



Copyright © Arcitura Education

Risks & Challenges

- Limited Portability Between Cloud Providers

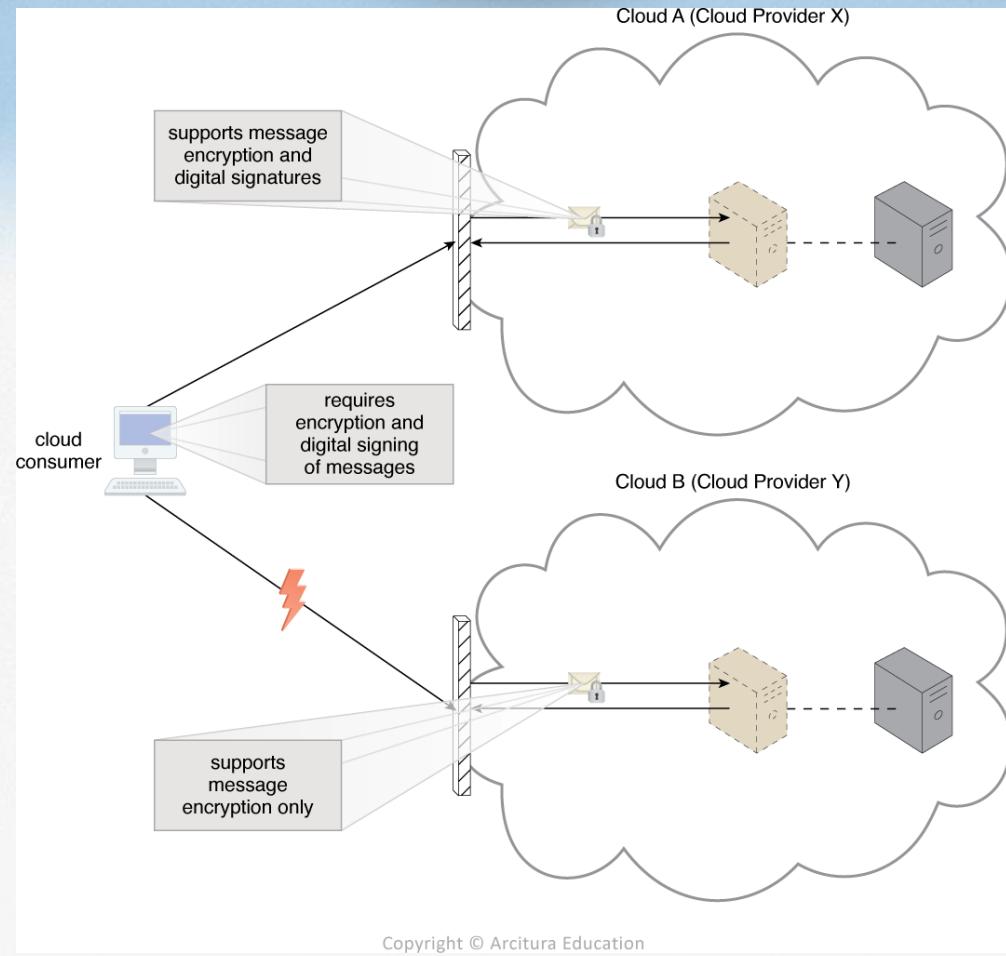


Figure: A cloud consumers application has a decreased level of portability when accessing a potential migration from Cloud A to Cloud B, because the cloud provider of Cloud B does not support the same security technologies as Cloud A.

Risks & Challenges

- Multiregional Compliance and Legal Issues

Summary of Key Points

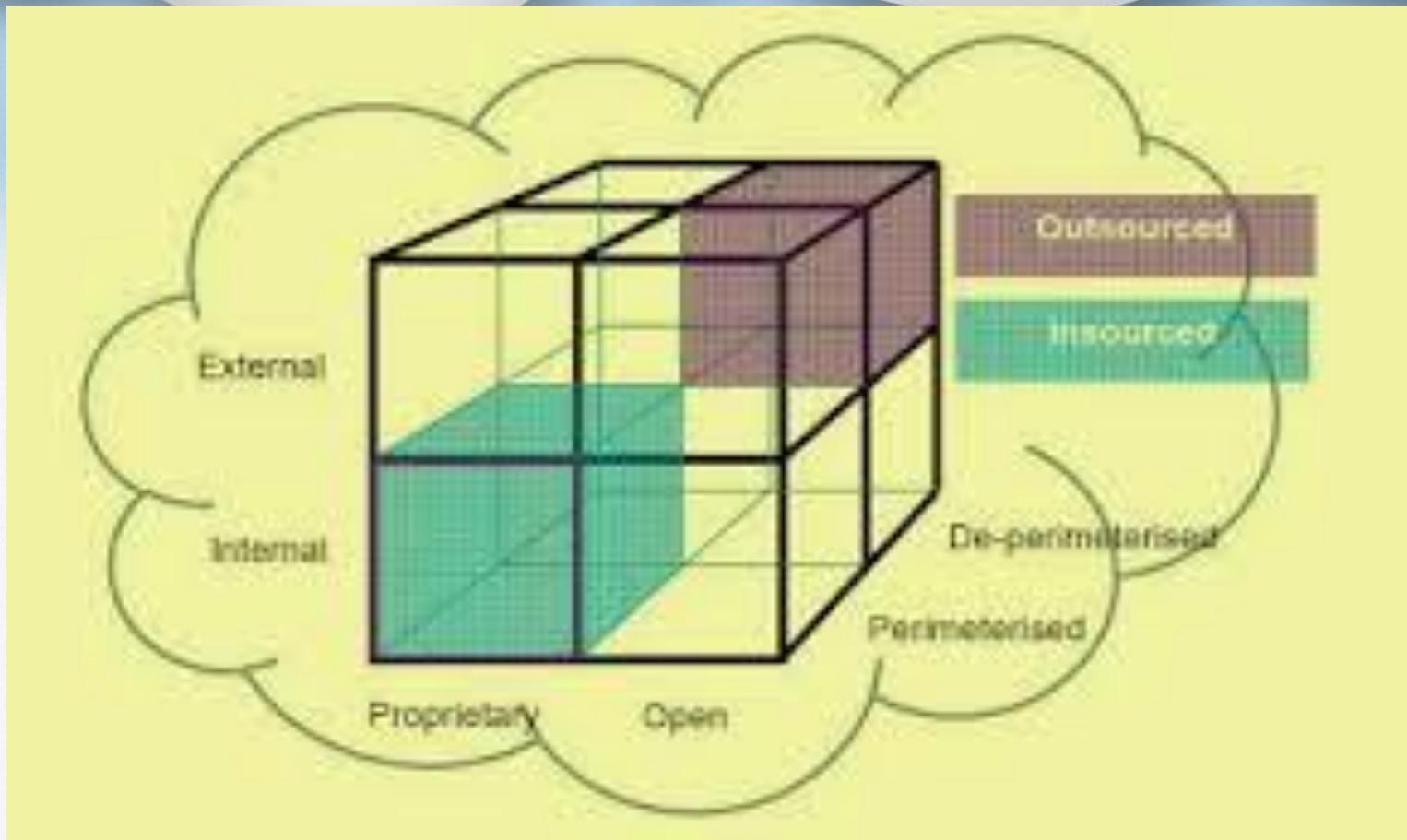
- Cloud environments can introduce distinct security challenges, some of which pertain to overlapping trust boundaries imposed by a cloud provider sharing IT resources with multiple cloud consumers.
- A cloud consumer's operational governance can be limited within cloud environments due to the control exercised by a cloud provider over its platforms.
- The portability of cloud-based IT resources can be inhibited by dependencies upon proprietary characteristics imposed by a cloud.
- The geographical location of data and IT resources can be out of cloud consumer's control when hosted by a third-party cloud provider. This can introduce various legal and regulatory compliance concern.

Muddiest Point

Cloud Characteristics

- The following six specific characteristic are common to the majority of cloud environments:
 - 1) On-demand usage
 - 2) Ubiquitous access
 - 3) Multitenancy (and resource pooling)
 - 4) Elasticity
 - 5) Measured usage
 - 6) Resiliency

Cloud Cube Model



Cloud Cube Model

- The Jericho Forum has designed the Cloud Cube Model to help select cloud formations for security cooperation.
- Their fascinating new cloud model helps IT managers and business tycoons assess the benefits of cloud computing.
- The Cloud Cube Model looks at the several different "cloud formations".
- They amount to the cloud service and deployment models.
- The sourcing dimension addresses the delivery of service.
- The Cloud Cube Model may be designed to let users show that the traditional notion of network ranges & its boundaries with network firewall no longer applies in Cloud computing.

Cloud Cube Model

- **Cloud Cube Model**, designed and developed by **Jericho forum**.
- Which helps to categorize the cloud network based on the four-dimensional factor: Internal/External, Proprietary/Open, De-Perimeterized/Perimeterized, and Insourced/Outsourced.

• **Dimension 1: Internal/External**

- This dimension defines the physical location of the data; where does the cloud form exist – inside or outside organization boundaries? If the cloud form is within the organization's physical boundaries, then it is internal.
- If it is outside the organization's physical boundaries, then it is external. It's important to note that the assumption that internal is necessarily more secure than external is false. The most secure usage model is the effective use of both internal and external cloud forms.

ii. Proprietary/Open

- The second type of cloud formation is **proprietary and open**. The proprietary or open dimension states about the state of ownership of the **cloud technology** and interfaces. It also tells the degree of interoperability, while enabling data transportability between the system and other cloud forms..
- The **proprietary dimension** means, that the organization providing the **service is securing** and protecting the data under their ownership.
- The **open dimension** is using a technology in which there are more suppliers. Moreover, the user is not constrained in being able to share the data and collaborate with selected partners using the open technology.ther cloud forms.

iii. De-Perimeterized / Perimeterized

- The third type of cloud formation is **De-perimeterized and Perimeterized**.
- To reach this form, the user needs collaboration oriented architecture and Jericho forum commandments.
The Perimeterised and De-perimeterized dimension tells us whether you are operating inside your traditional it mindset or outside it.
- **Perimeterized dimension** means, continuing to operate within the traditional it boundary, orphan signaled by network firewalls. With the help of VPN and operation of the virtual server in your own IP domain, the user can extend the organizations perimeter into external Cloud Computing domain. This means that the user is making use of the own services to control access.

iii. De-Perimeterized / Perimeterized

- **De-perimeterized dimension** means the system perimeter is architected on the principles outlined in the Jericho forums commandments. In De-perimeterized dimension, the data will be encapsulated with metadata and mechanisms, which will further help to protect the data and limit the inappropriate usage.

iv. Insourced/Outsourced

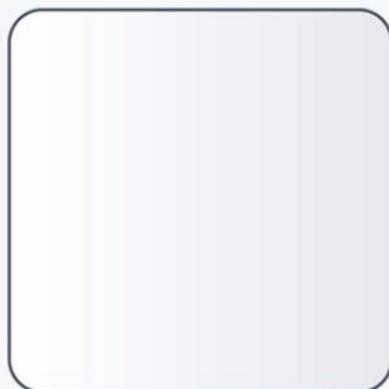
- The **Insourced and outsourced dimensions** have two states in each of the eight cloud forms. In the *outsourced dimension* the services provided by the third party, whereas in the *insourced dimension* the services provided by the own staff under the control.
- In this few organizations that are traditional bandwidth software or hardware, providers will run fluently on becoming cloud service providers.
- The organizations which are seeking to procedure cloud services must have the ability to set legally binding collaboration agreement. In this, an organization should ensure that data is deleted from the service provider's Infrastructure.

Questions For Cloud Cube Model

- The Jericho forum states that there are three key questions, which a customer should ask their **Cloud Computing** supplier. So, that they must be aware that the data is secure and protected. The three questions are-
- Q 1. Wherein the cloud cube model is the cloud supplier operating while providing the services?
- Q 2. How will the clouds suppliers get a surety when the customer is using services in a cloud from that has maintained the features as per the expectations?
- Q 3. How can a customer ensure that the data which is stored in the cloud services will be available at the time of mishappenings such as bankruptcy or change in business direction?

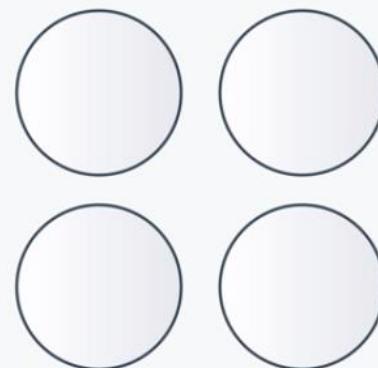
MONOLITHIC, SOA & MICROSERVICES

Monolithic vs. SOA vs. Microservices



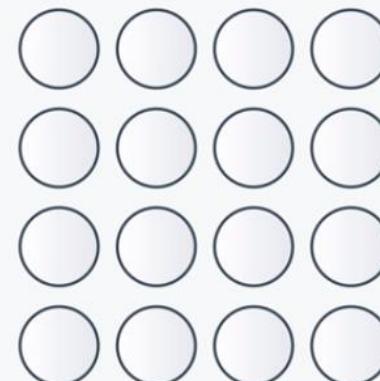
Monolithic

Single Unit



SOA

Coarse-grained



Microservices

Fine-grained

Activate Windows
Go to Settings to activate Windows.



SOA Vs Microservice



SOA is like an orchestra where each artist is performing with his/her instrument while the music director guides them all.

With Microservices each dancer is independent and know what they need to do. If they miss some steps they know how to get back on the sequence.

Activate Windows
Go to Settings to activate Windows.

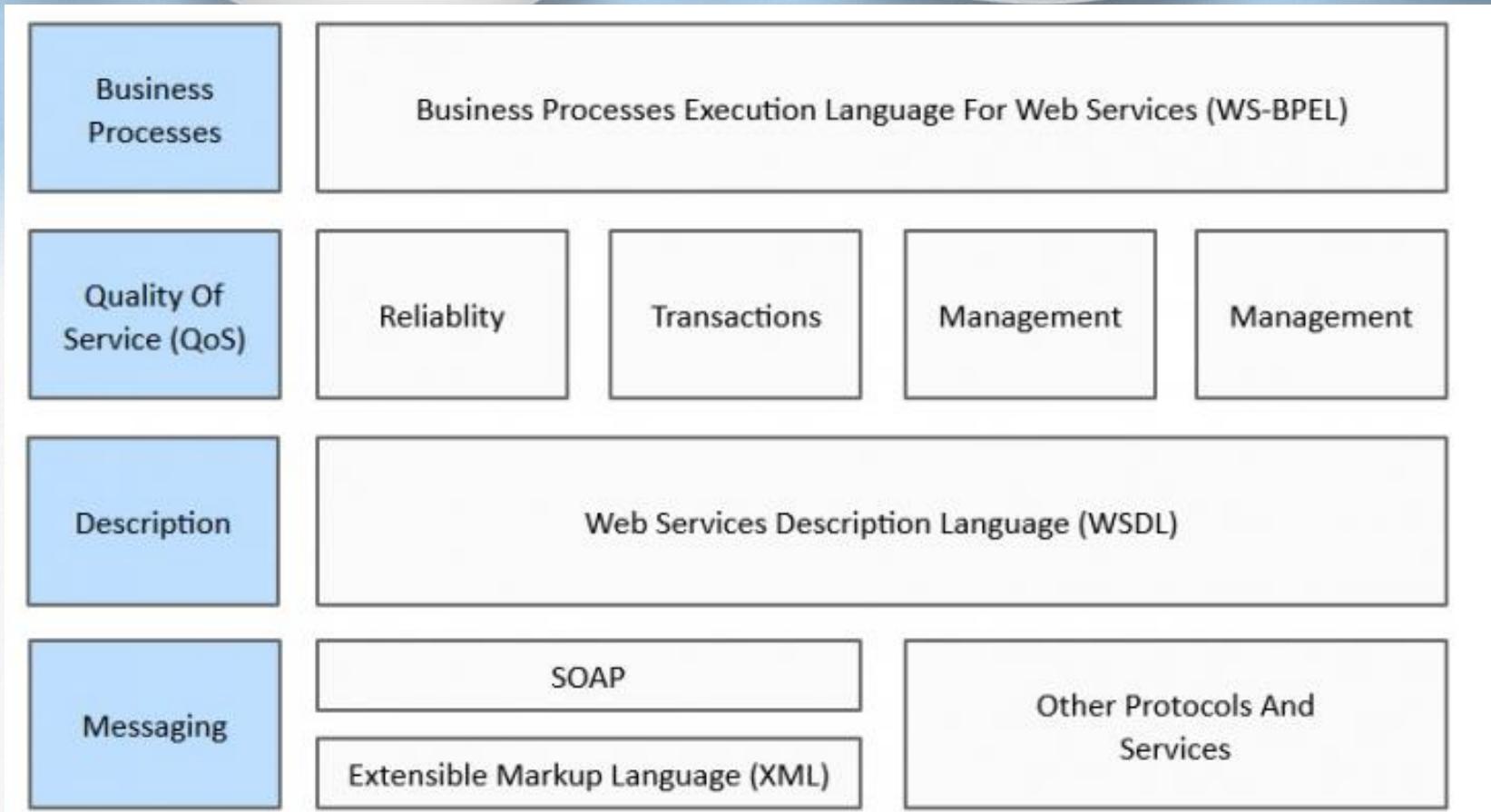
Service Oriented Architecture

- **Service-oriented architecture (SOA)** is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.
- A SOA service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online. SOA is also intended to be independent of vendors, products and technologies.

Service Oriented Architecture

- A service has four properties according to one of many definitions of SOA:
 - It logically represents a business activity with a specified outcome.
 - It is self-contained.
 - It is a black box for its consumers, meaning the consumer does not have to be aware of the service's inner workings.
 - It may consist of other underlying services.
- Different services can be used in conjunction to provide the functionality of a large software application, a principle SOA shares with modular programming.

SOA Architecture and Protocols



Service Oriented Architecture

- SOA architecture is viewed as five horizontal layers. These are described below:
 - 1) **Consumer Interface Layer:** These are GUI based apps for end users accessing the applications.
 - 2) **Business Process Layer:** These are business-use cases in terms of application.
 - 3) **Services Layer:** These are whole-enterprise, in service inventory.
 - 4) **Service Component Layer:** are used to build the services, such as functional and technical libraries.
 - 5) **Operational Systems Layer:** It contains the data model.

Service Oriented Architecture

- A manifesto was published for service-oriented architecture in October, 2009. This came up with six core values which are listed as follows:
 - 1) **Business value** is given more importance than technical strategy.
 - 2) **Strategic goals** are given more importance than project-specific benefits.
 - 3) **Intrinsic interoperability** is given more importance than custom integration.
 - 4) **Shared services** are given more importance than specific-purpose implementations.
 - 5) **Flexibility** is given more importance than optimization.
 - 6) **Evolutionary refinement** is given more importance than pursuit of initial perfection.

Guiding Principles of SOA:

- **Standardized service contract:** Specified through one or more service description documents.
- **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
- **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
- **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.

Guiding Principles of SOA:

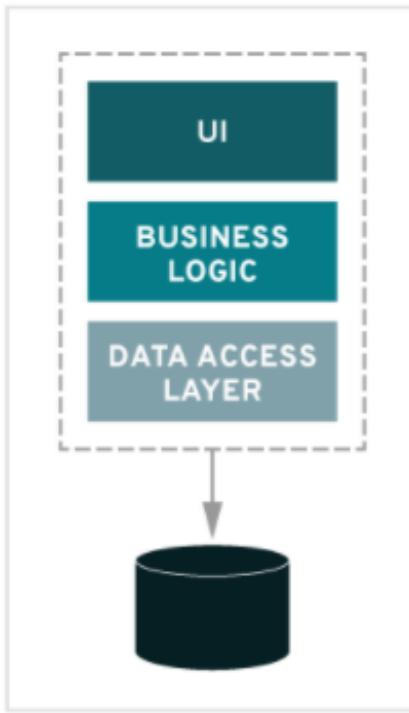
- **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
- **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
- **Composability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

Microservices

- **Microservice** architecture – a variant of the service-oriented architecture (SOA) structural style – arranges an application as a collection of loosely coupled services.
- In a microservices architecture, services are fine-grained and the protocols are lightweight.

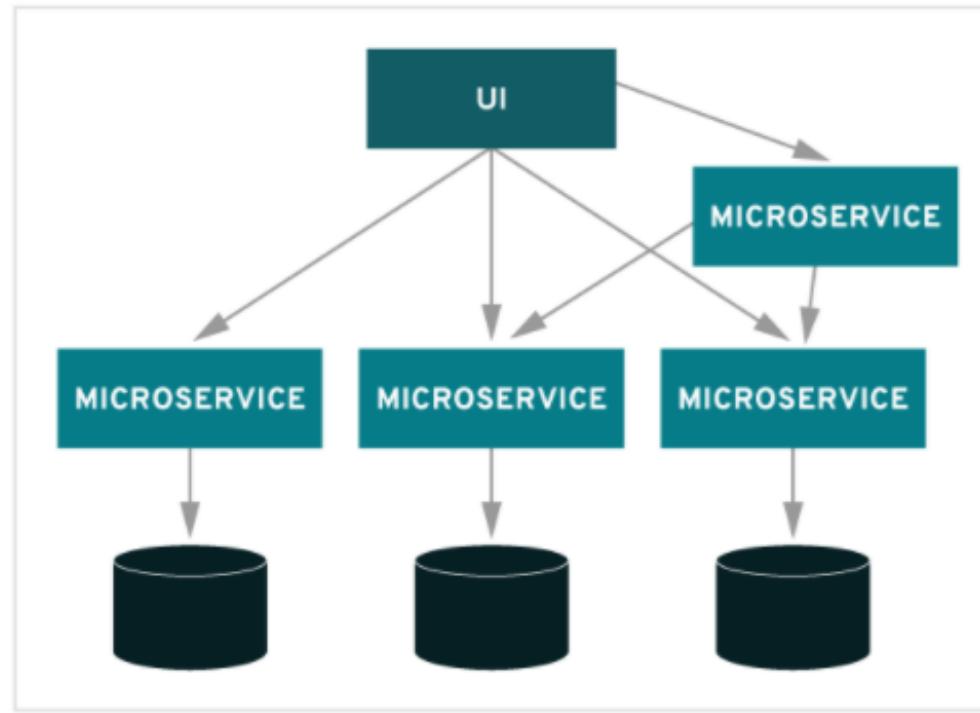
Microservices

MONOLITHIC



MICROSERVICES

VS.



Microservices Features



WEB 1.0 To 5.0

- **Web 0.0** – Developing the internet
- **Web 1.0** – The shopping carts & static web
- **Web 2.0** – The writing and participating web
- **Web 3.0** – The semantic executing web
- **Web 4.0** – “Mobile Web”
- **Web 5.0**- Open, Linked and Intelligent
Web = Emotional Web. “The next web”

THANK YOU!!!!

Presentation Topic

Unit IV - Amazon Web Services

Vishal Ambadas Meshram

vishal.meshram@viit.ac.in

Department of Computer Engineering



BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)





Cloud Computing:

The practice of using network of remote servers hosted on the Internet to store, manage, and process data rather than a local server or a personal computer.

Cloud Concept



On-Premise

1. You own the Server.
2. You hire IT people.
3. You pay or rent the real-estate.
4. You take all the risk.

Cloud



1. Someone else owns the Server.
2. Someone else hire IT people.
3. Someone else pay or rent the real-estate.
4. You are responsible for configuring your cloud services & code.
Someone else takes care of the rest.

Cloud Computing Models

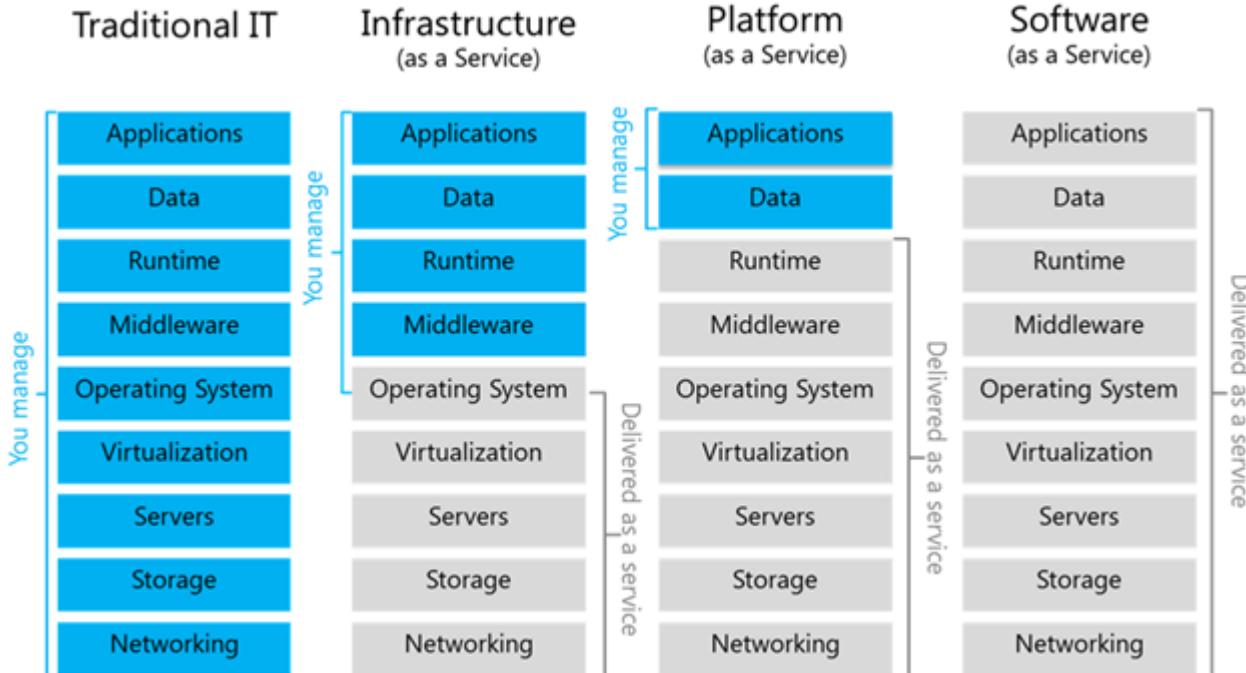


Fig Ref : <https://dachou.github.io/2018/09/28/cloud-service-models.html>

Cloud Computing Deployment Models

Cloud

Fully utilizing cloud computing

Hybrid

Using both Cloud and On-Premise

On-Premise

Deploying resources on-premises, using virtualization and resource management tools, is sometimes called “private cloud”.



- Startups
- SaaS offerings
- New projects and companies



- Banks
- FinTech, Investment Management
- Large Professional Service providers
- Legacy on-premise



- Public Sector eg. Government ACTIVATE WINDOWS
Go to Settings to activate Windows.
- Super Sensitive Data eg. Hospitals
- Large Enterprise with heavy regulation eg. Insurance Companies

Contents:

- **What is AWS**
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

What is AWS?

aws

Link: https://www.youtube.com/watch?v=a9__D53WsUs

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

AWS Global Infrastructure Map

AWS serves over a million active customers in more than **190 countries**. AWS now spans **77 Availability Zones** within **24 geographic regions** around the world and has announced plans for 18 more Availability Zones and 6 more AWS Regions in Australia, India, Indonesia, Japan, Spain, and Switzerland.



AWS Global Infrastructure Map

Regions

- Physical Location in the world with multiple Availability Zones (AZ's).
- It's a geographical distinct location.
- Every region is physically isolated from and independent of every other region in terms of location, power, water supply.
- Each region has at least 2 AZ's
- AWS largest region is "**US-EAST**" (**Northern Virginia**)
- US-EAST1 is the region where you see all your billing information.

Availability Zones

- Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities.
- AZ's offer you the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center.
- AZ's are represented by region code, followed by a letter identifier. Ex. "us-east-1a".
- "MultiAZ", distributing your instance across multiple AZ's allows failover configuration for handling request when goes down.

Edge Locations

- **An Edge location is a data center owned by a trusted partner of AWS which has a direct connection to the AWS network.**
- These location serve requests for CloudFront & Route 53. Requests going to either of these services will be routed to the nearest edge location automatically.
- S3 transfer acceleration traffic & API Gateway endpoint traffic also use the AWS Edge network.

“Customers are increasingly choosing AWS to host their cloud-based infrastructure and realize increased performance, security, reliability, and scale wherever they go. For the tenth year in a row, AWS is evaluated as a Leader in the [2020 Gartner Magic Quadrant for Cloud Infrastructure and Platform Services](#), placed highest in both axes of measurement—Ability to Execute and Completeness of Vision—among the top 7 vendors named in the report.”



Figure 1. Magic Quadrant for Cloud Infrastructure and Platform Services



Gartner, Magic Quadrant for Cloud Infrastructure & Platform Services, Raj Bala, Bob Gill, Dennis Smith, David Wright, Kevin Ji, 1 September 2020. This graphic was published by Gartner, Inc. as part of a larger research document and should be evaluated in the context of the entire document. The Gartner document is available upon request from AWS. Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

Lower Costs with AWS Up-Front and Increase Savings as Your Usage Grows

1

Replace up-front capital expense with low variable cost

“Average of 400 servers replaced per customer”

2

Economies of scale have allowed us to consistently lower costs

44 Price Reductions

3

Pricing model choice to support variable & stable workloads

On-demand
Reserved
Spot

4

Save more money as you grow bigger

Tiered Pricing
Volume Discounts



Source: IDC Whitepaper, sponsored by Amazon, “The Business Value of Amazon Web Services Accelerates Over Time.” July 2012

4X More Reliable & 1/4 the Cost of On-Premises Infrastructure

END USERS BENEFITED FROM FEWER SERVICE DISRUPTIONS AND QUICKER RECOVERY ON AMAZON CLOUD INFRASTRUCTURE,
REDUCING DOWNTIME BY 72%

AMAZON CLOUD INFRASTRUCTURE REPRESENTS A
70% SAVINGS COMPARED WITH ON-PREMISE SOLUTIONS



WHITE PAPER

The Business Value of Amazon Web Services Accelerates Over Time

Sponsored by: Amazon

Randy Perry Stephen D. Hendrick
July 2012

EXECUTIVE SUMMARY

In early 2012, IDC interviewed 11 organizations that deployed applications on Amazon cloud infrastructure services. The purpose of the IDC analysis was to understand the economic impact of Amazon cloud infrastructure services over time, beyond the well-documented benefit of reduction in capex and opex. Specifically,

IDC set out to understand the long-term economic implications of moving workloads onto Amazon cloud infrastructure services, the impact of moving applications on developer productivity and business agility, and the new opportunities that businesses could address by moving resources onto Amazon cloud infrastructure services. The organizations interviewed ranged from small and medium-sized companies to companies with as many as 160,000 employees. Organizations in our study had been Amazon Web Services (AWS) customers for as few as seven months to as many as 5.3 years. Our interviews were designed to elicit both quantifiable information and anecdotes so that IDC could interpret the full return-on-investment (ROI) impact of Amazon cloud infrastructure services on these organizations. The study represents a broad range of

Business Value Highlights: Applications Running on AWS

- Five-year ROI: 625%
- Payback period: 7.1 months
- Software development productivity increase: 507%
- Average savings per application: \$518,990
- Downtime reduction: 72%
- IT productivity increase: 52%
- Five-year TCO savings: 70%



Architected for Enterprise Security Requirements

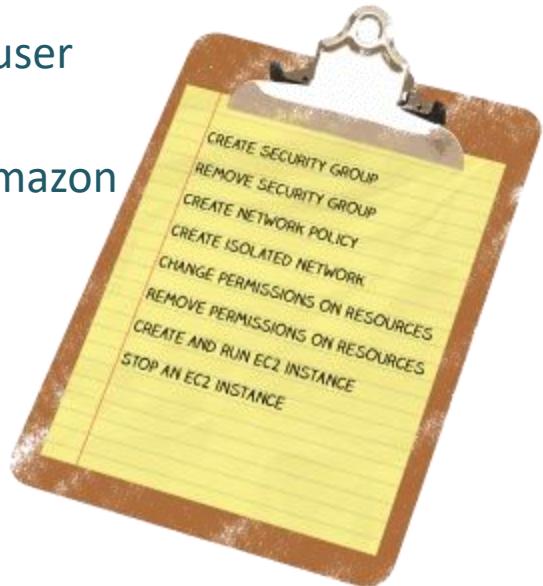
Certifications and accreditations for workloads that matter



AWS CloudTrail - AWS API call logging for governance & compliance

Log and review user activity

Stores data in Amazon S3, or archive to Amazon Glacier



Increased agility has become the
#1 reason businesses use the AWS
cloud



Enterprises Can't Afford to be Slow

Old World: Infrastructure in Weeks



AWS: Infrastructure in Minutes



- Add New Dev Environment
- Add New Prod Environment
- Add New Environment in Japan
- Add 1,000 Servers
- Remove 1,000 servers
- Deploy 2 PB Data warehouse
- Shut down 2 PB Data warehouse

Everything changes with this kind of agility

A Culture of Innovation: Experiment Often and Fail Without Risk



On-Premises

Experiment Infrequently

Failure is expensive

Less Innovation



Experiment Often

Fail quickly at a low cost

More Innovation

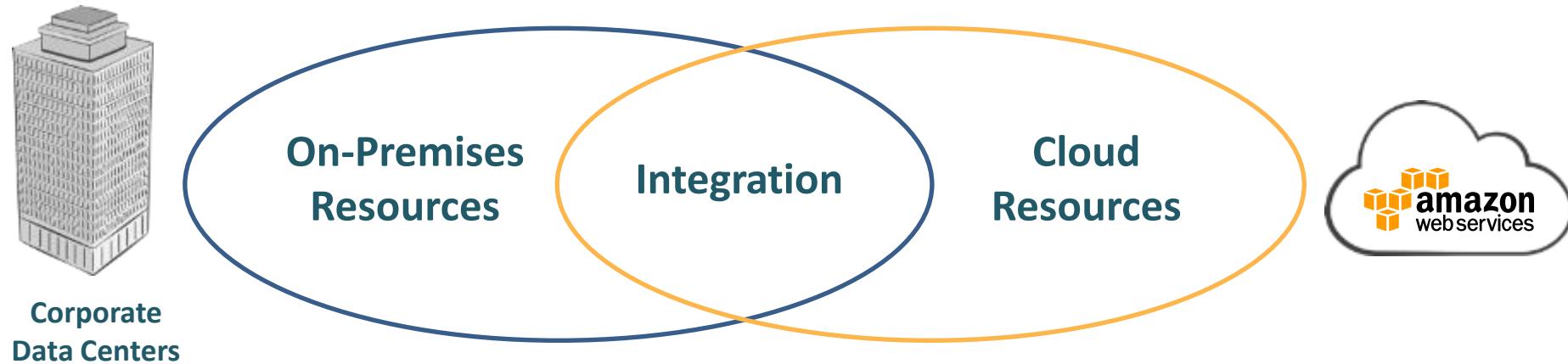
Many Enterprises Worry That These are the Only Two Choices

Build a
“private”
cloud

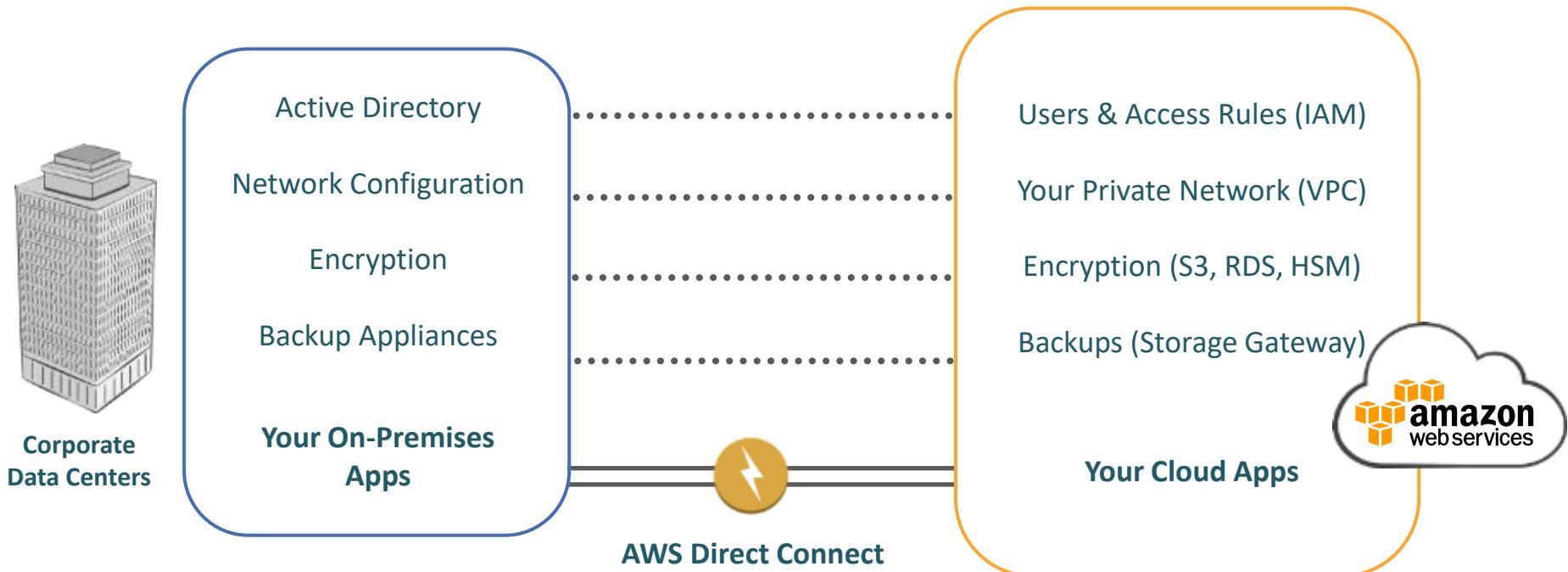


Rip everything out
and move to
AWS

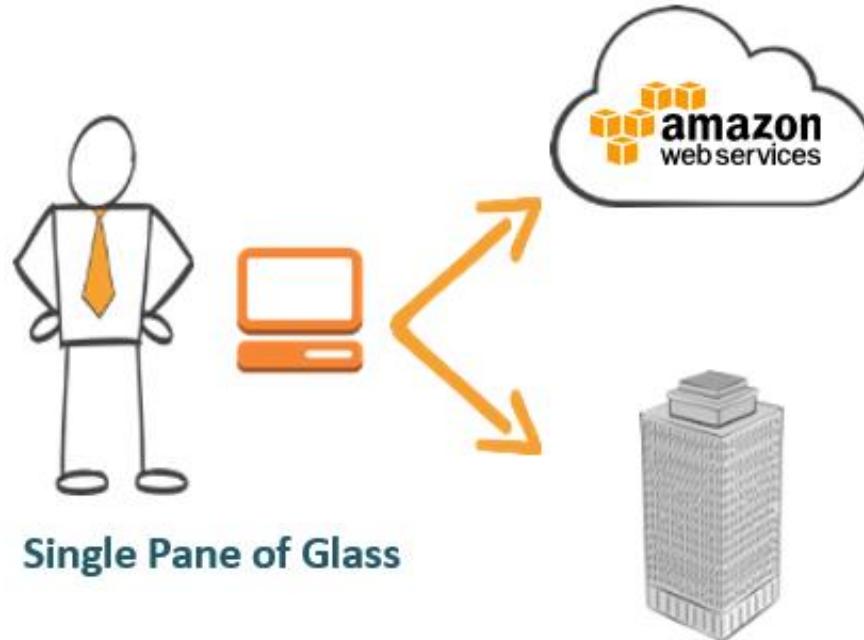
The Good News is that Cloud isn't an 'All or Nothing' Choice



Integrating AWS with Your Existing On-Premises Infrastructure



Tools to Help Customers Manage Resources Across Environments



Enterprises Use Cases on AWS

**Enterprise Apps
and Dev./Test**

**Big Data and
HPC**

**Storage, Backup
and Archival**

**Web, Mobile, and
Social Apps**

**Disaster
Recovery**

**Virtual
Desktops**

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- **AWS Account setup and billing alarms**
- Compute services
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- IAM

AWS billing alarms

1. Billing Preferences

2. Budget

3. Cloud Watch

Spend Summary Cost Explorer

Welcome to the AWS Billing & Cost Management console. Your last month, month-to-date, and month-end forecasted costs appear below.

Current month-to-date balance for February 2021, the exchange rate for the Payment Currency is estimated.

0.00 USD which converts to

0.00 INR

at today's exchange rate of 73.48255

Bills

Amount
\$1.6
\$1.59
\$1.2
\$0.8

Preferences

Billing preferences

Month-to-Date Spend by Service

The chart below shows the proportion of total monthly spend by service.

My Account 933126026785

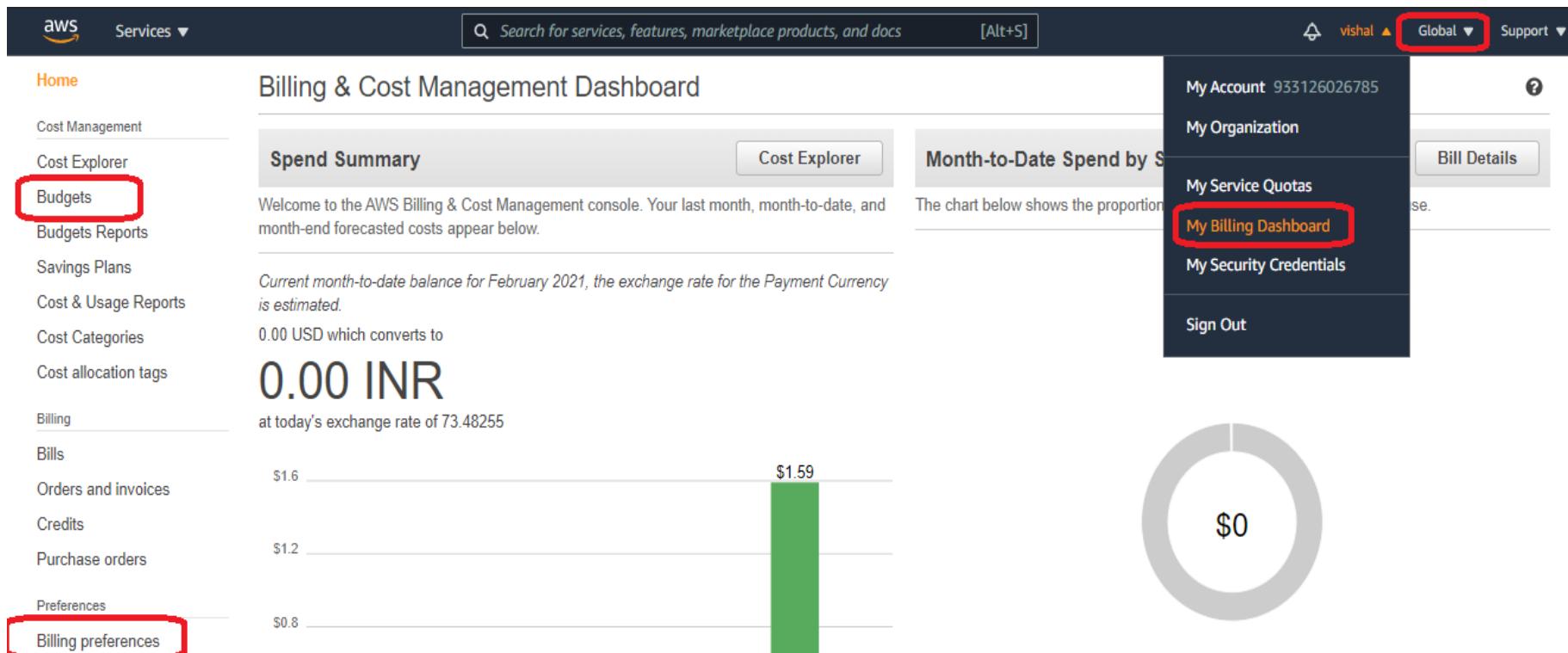
My Organization

My Service Quotas

My Billing Dashboard

My Security Credentials

Sign Out



Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- **IAM Introduction**
- Compute services (EC2)
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

IAM Introduction

- IAM stands for Identity Management Service.
- IAM is global service.
- Instead of using root account create a IAM user.
- Never share your root or IAM credentials.
- Policies are written in JSON (JavaScript Object Notation)

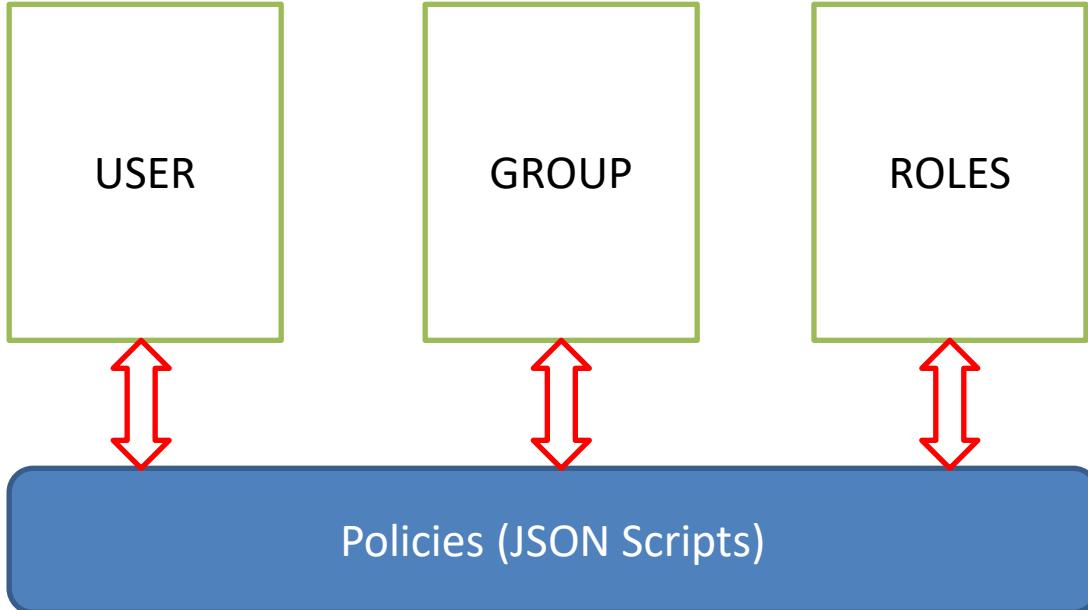
The screenshot shows the AWS IAM dashboard. On the left, a sidebar menu is open, with the 'Identity and Access Management (IAM)' section highlighted by a red box. The main content area displays the 'IAM dashboard' with the following sections:

- Sign-in URL for IAM users in this account:** <https://viit.signin.aws.amazon.com/console> (with Edit and Delete alias options)
- IAM resources:**
 - Users: 1
 - Groups: 1
 - Customer managed policies: 0
 - Roles: 6
 - Identity providers: 0
- Security alerts:** A warning message: "The root user for this account does not have Multi-factor authentication (MFA) enabled. Enable MFA to improve security for this account."
- Best practices:**
 - Grant least privilege access: Establishing a principle of least privilege ensures that identities are only permitted to perform the most minimal set of functions necessary to fulfill a specific task, while balancing usability and efficiency.
 - Use AWS Organizations: Centrally manage and govern your environment as you scale your AWS resources. Easily create new AWS accounts, group accounts to organize your workflows, and apply policies to accounts or groups for governance.
 - Enable Identity federation: Manage users and access across multiple services from your preferred identity source. Using AWS Single Sign-On centrally manage access to multiple AWS accounts and provide users with single sign-on access to all their assigned accounts from one place.
 - Enable MFA: For extra security, we recommend that you require multi-factor authentication (MFA) for all users.

On the right side, there are additional links and tools:

- Additional information: IAM documentation, Videos, IAM release history and additional resources.
- Tools: Web identity federation playground, Policy simulator.
- Quick links: My access key.
- Related services: AWS Organizations, AWS Single Sign-on (SSO).

AWS account ID:
933126026785



The screenshot shows the AWS IAM Security Credentials page. The left sidebar lists various IAM management options like Dashboard, Access management, Groups, Users, Roles, Policies, Identity providers, Account settings, and Access reports. A search bar at the top has 'iam' typed into it. The main content area is titled 'Your Security Credentials' and provides instructions for managing AWS credentials. It highlights 'Password' and 'Multi-factor authentication (MFA)' with red boxes. A vertical navigation bar on the right includes links for My Account (with account number), My Organization, My Service Quotas, My Billing Dashboard, and My Security Credentials (which is also highlighted with a red box). The 'Sign Out' link is at the bottom of this bar. The top right corner shows a user profile for 'vishal' and global account information.

aws Services ▾

Search iam

vishal Global ▾ Support ▾

Identity and Access Management (IAM)

Dashboard

Access management

Groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Search IAM

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, see [Managing IAM User Credentials](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

▼ Password

You use an email address and password to sign in to secure pages on AWS, such as the AWS Management Console, AWS Forum, and AWS CLI. Create a password that contains many characters, including numbers and punctuation. Store your password securely, do not share it, and do not reuse it.

[Click here](#) to change the password, name, or email address for your root AWS account.

▲ Multi-factor authentication (MFA)

▲ Access keys (access key ID and secret access key)

▲ CloudFront key pairs

▲ X.509 certificate

▲ Account identifiers

My Account 933126026785

My Organization

My Service Quotas

My Billing Dashboard

My Security Credentials

Sign Out

For MFA: Install “Google Authenticator” App.

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- **Compute services (EC2)**
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

EC2 (Elastic Compute Cloud)

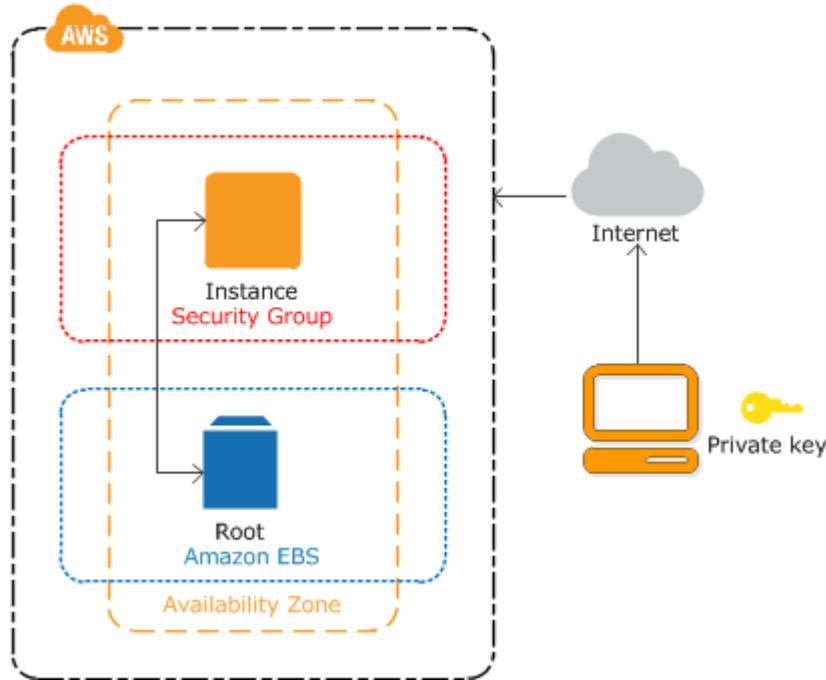
- ❖ EC2 stands for Elastic Compute Cloud.
- ❖ EC2 is a virtual machine on AWS.
- ❖ As per amazon “Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment.”



Get started with Amazon EC2 Linux instances



- Overview



Steps:

- Step 1: Launch an instance
- Step 2: Connect to your instance
- Step 3: Clean up your instance

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 1: To launch an instance

- 1) Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
- 2) From the console dashboard, choose **Launch Instance**.
- 3) The **Choose an Amazon Machine Image (AMI)** page displays a list of basic configurations, called *Amazon Machine Images (AMIs)*, that serve as templates for your instance. Select an HVM version of Amazon Linux 2. Notice that these AMIs are marked "Free tier eligible."
- 4) On the **Choose an Instance Type** page, you can select the hardware configuration of your instance. Select the t2.micro instance type, which is selected by default. The t2.micro instance type is eligible for the free tier. In Regions where t2.micro is unavailable, you can use a t3.micro instance under the free tier. For more information, see [AWS Free Tier](#).
- 5) Choose **Review and Launch** to let the wizard complete the other configuration settings for you.
- 6) On the **Review Instance Launch** page, under **Security Groups**, you'll see that the wizard created and selected a security group for you. You can use this security group, or alternatively you can select the security group that you created when getting set up using the following steps:
 - a) Choose **Edit security groups**.
 - b) On the **Configure Security Group** page, ensure that **Select an existing security group** is selected.
 - c) Select your security group from the list of existing security groups, and then choose **Review and Launch**.
- 7) On the **Review Instance Launch** page, choose **Launch**.
- 8) When prompted for a key pair, select **Choose an existing key pair**, then select the key pair that you created when getting set up.
- 9) A confirmation page lets you know that your instance is launching. Choose **View Instances** to close the confirmation page and return to the console.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

The screenshot shows the AWS EC2 Dashboard. On the left sidebar, under the 'Instances' section, the 'Launch Instance' button is highlighted with a red box. In the main content area, the 'Resources' section displays various metrics: Instances (running) 0, Dedicated Hosts 0, Elastic IPs 0, Instances 2, Key pairs 1, Load balancers 1, Placement groups 0, Security groups 3, Snapshots 1, and Volumes 0. A tooltip message encourages using the AWS Launch Wizard for Microsoft SQL Server. Below the resources, there's a 'Launch instance' section with a red box around the 'Launch instance' button, which is described as launching an Amazon EC2 instance, a virtual server in the cloud. To the right, the 'Service health' section shows the region as Asia Pacific (Mumbai) and the status as 'This service is operating'. The top right corner shows account attributes for vishal in Mumbai.

aws Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

New EC2 Experience Tell us what you think

EC2 Dashboard New

Events

Tags

Limits

Instances

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

Elastic Block Store

Volumes

Resources

You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	0	Dedicated Hosts	0
Elastic IPs	0	Instances	2
Key pairs	1	Load balancers	1
Placement groups	0	Security groups	3
Snapshots	1	Volumes	0

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more X

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▾

Service health

C Service Health Dashboard

Region Status

Asia Pacific (Mumbai) This service is operating

Account attributes

Supported platforms

- VPC

Default VPC vpc-30308859

Settings

EBS encryption

Zones

Default credit specification

Console experiments

Explore AWS

Enable Best Price-Performance with AWS Graviton2

AWS Graviton2 powered EC2 instances enable up to 40% better price performance for a broad spectrum of cloud workloads. [Learn more](#)

Get Up to 40% Better Price

vishal Mumbai

1. Search for “ec2” service.
2. Click on “Launch Instance”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs

Free tier only ⓘ

AMI Name	Description	Root device type	Virtualization type	ENI Enabled	Action
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-08e0ca9924195beba (64-bit x86) / ami-0437d5dbe8fdc3d52 (64-bit Arm)	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0a9d27a9f4f5c0efc (64-bit x86) / ami-0816d75a127c17a49 (64-bit Arm)	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0b3acf3edf2397475 (64-bit x86) / ami-0ab71076ab9b53b0d (64-bit Arm)	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-073c8c0760395aab8 (64-bit x86) / ami-029dbbe5a11f53cf7 (64-bit Arm)	Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	ebs	hvm	Yes	Select <input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)

3. Select “Amazon Linux 2 AMI (HVM).
4. Click on Select.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, ~ 1 GiB memory, EBS only)

	Family	Type	vCPUs ⓘ	Memory (GiB) ⓘ	Instance Storage (GB) ⓘ	EBS-Optimized Available ⓘ	Network Performance ⓘ	IPv6 Support ⓘ
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel

Previous

Review and Launch

Next/Configure Instance Details

3. Select “t2.micro”
4. Click on “Configure Instance Details”.

Services ▾ Search for services, features, marketplace products, and docs [All 13]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances Launch into Auto Scaling Group

Purchasing option Request Spot instances

Network vpc-30308859 (default) Create new VPC

Subnet No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP No preference (default subnet in any Availability Zone)

subnet-af65abd4 | Default in ap-south-1c
subnet-db9613b2 | Default in ap-south-1a
subnet-883cc7c5 | Default in ap-south-1b

Placement group

Capacity Reservation Open

Domain join directory No directory Create new directory

IAM role None Create new IAM role

CPU options Specify CPU options

Shutdown behavior Stop

Stop - Hibernate behavior Enable hibernation as an additional stop behavior

Cancel Previous Review and Launch Next: Add Storage

5. Click on “Add Storage”.

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type <i>i</i>	Device <i>i</i>	Snapshot <i>i</i>	Size (GiB) <i>i</i>	Volume Type <i>i</i>	IOPS <i>i</i>	Throughput (MB/s) <i>i</i>	Delete on Termination <i>i</i>	Encryption <i>i</i>
Root	/dev/xvda	snap-07e0efc01c68d3978	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

5. Click on “Add Tags”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ⓘ	Volumes ⓘ	Network Interfaces ⓘ
Name	My-First-Instance	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Cancel

Previous

Review and Launch

Next: Configure Security Group

5. Click on “Configure Security Group”.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group

Select an existing security group

Security group name:

My-SG

Description: launch-wizard-1 created 2021-02-14T17:27:05.680+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	allow traffic from all IP addresses

Add Rule



Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous **Review and Launch**
Go to Settings to activate Windows.

5. Click on “Review and launch”.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

 Improve your instances' security. Your security group, My-SG, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details

[Edit AMI](#)**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-08e0ca9924195beba****Free tier eligible**

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...

Root Device Type: ebs Virtualization type: hvm

Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups

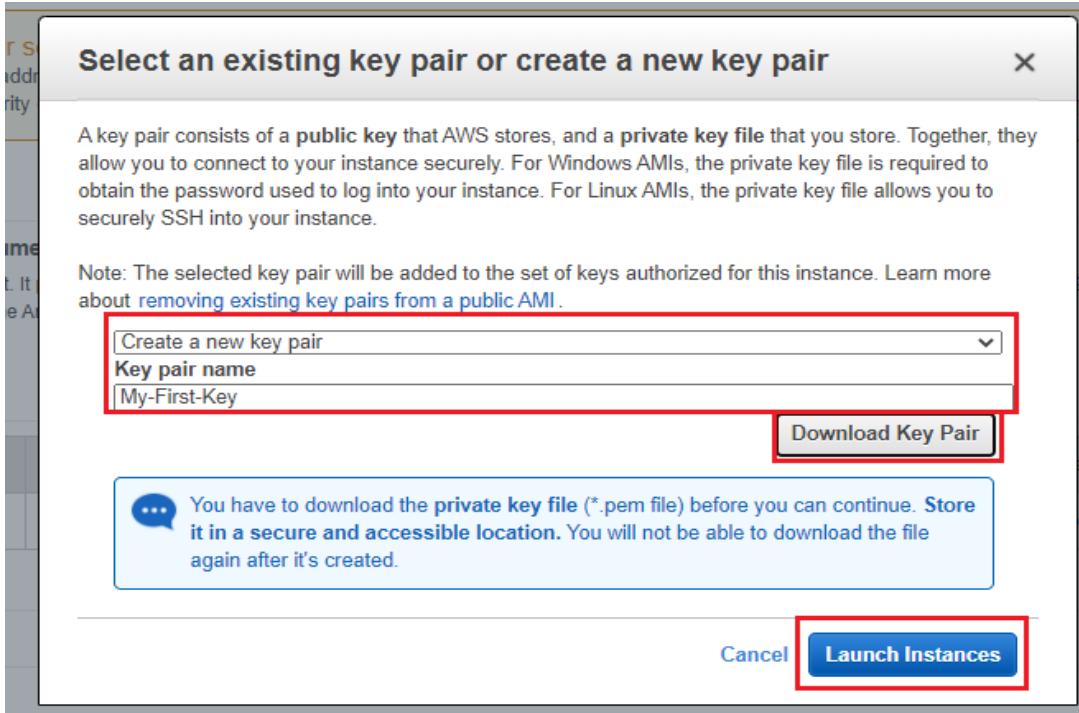
[Edit security groups](#)

Security group name: My-SG

Activate | Deactivate | [Cancel](#) | [Previous](#)

Launch

5. Click on “Launch”.



5. Click on “Create a new key pair”.
6. Download Key Pair.
7. Click on “Launch Instance”.
8. Click on “View Instance”.

The screenshot shows the AWS EC2 Instances page. At the top, there's a header with 'Instances (1/3)' and 'Info' buttons, followed by 'Connect', 'Instance state', 'Actions', and 'Launch instances' buttons. Below the header is a search bar with 'Filter instances' placeholder text and navigation arrows. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
First	i-0610ff454f83aca5b	Terminating	t2.micro	-	No alarms	ap-south-1a	-
Second	i-0da7e305183d1c90a	Terminating	t2.micro	-	No alarms	ap-south-1b	-
My-First-Inst...	i-091152db64fd104a5	Running	t2.micro	Initializing	No alarms	ap-south-1b	ec2-15

Below the table, a modal window is open for the selected instance 'My-First-Inst...'. The title is 'Instance: i-091152db64fd104a5 (My-First-Instance)'. The modal has tabs for 'Details', 'Security', 'Networking', 'Storage', 'Status checks', 'Monitoring', and 'Tags'. The 'Details' tab is active, showing sections for 'Instance summary' and 'Instance details'. The 'Platform' and 'AMI ID' fields are visible at the bottom.

9. Select your instance and check the Instance state, it must be “Running”.

Congratulations!!!!!!

Steps to create EC2 Instance.



Step 2: Connect to your instance

There are several ways to connect to your Linux instance. For more information, see [Connect to your Linux instance](#).

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like New EC2 Experience, EC2 Dashboard, Events, Tags, Limits, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and Elastic Block Store. The main area displays three instances: First, Second, and My-First-Inst... (selected). The 'My-First-Inst...' row has a context menu open, with 'Connect' highlighted. At the top of the page, there's another 'Connect' button highlighted with a red box.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
First	i-06f0ff454f83aca5b	Terminated	t2.micro	-	No alarms	ap-south-1a	-
Second	i-0da7e305183d1c90a	Terminated	t2.micro	-	No alarms	ap-south-1b	-
My-First-Inst...	i-091152db64fd104a5	Running	t2.micro	2/2 checks ...	No alarms	ap-south-1b	ec2-15-

1. Click on “Connect”.
2. Or right click on Instance and select “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.

Step 2: Connect to your instance



Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

EC2 Instance Connect **Session Manager** **SSH client**

Instance ID
 [i-091152db64fd104a5 \(My-First-Instance\)](#)

Public IP address
 [15.206.212.136](#)

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

1. There are 3 ways to connect your instance.
2. Click on “EC2 Instance Connect”.
3. Click on “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 2: Connect to your instance

Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

EC2 Instance Connect **Session Manager** **SSH client**

Instance ID

Public IP address

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

ⓘ Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

1. There are 3 ways to connect your instance.
2. Click on “EC2 Instance Connect”.
3. Click on “Connect”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

The screenshot shows a terminal window with the URL <https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5>. The terminal output shows:

```
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-14-195 ~]$ sudo su
[root@ip-172-31-14-195 ec2-user]# yum update -y
```

The screenshot shows a terminal window with the URL <https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5>. The terminal output shows several blank lines followed by:

```
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]#
root@ip-172-31-14-195 ec2-user]# sudo yum install -y httpd.x86_64
```

1. Give command -> sudo su and yum update -y
2. Sudo yum install -y httpd.x86_64 => to install Apache HTTP Server

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

```
← → C https://ap-south-1.console.aws.amazon.com/ec2/v2/connect/ec2-user/i-091152db64fd104a5
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ 
[ec2-user@ip-172-31-14-195 ~]$ sudo su
root@ip-172-31-14-195 ec2-user]# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
root@ip-172-31-14-195 ec2-user]# systemctl start httpd
root@ip-172-31-14-195 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2021-02-14 12:57:54 UTC; 6s ago
     Docs: man:httpd.service(8)
 Main PID: 4950 (httpd)
   Status: "Processing requests..."
  CGroup: /system.slice/httpd.service
          ├─4950 /usr/sbin/httpd -DFOREGROUND
          ├─4951 /usr/sbin/httpd -DFOREGROUND
          ├─4952 /usr/sbin/httpd -DFOREGROUND
          ├─4953 /usr/sbin/httpd -DFOREGROUND
          ├─4954 /usr/sbin/httpd -DFOREGROUND
          └─4955 /usr/sbin/httpd -DFOREGROUND
Feb 14 12:57:54 ip-172-31-14-195.ap-south-1.compute.internal systemd[1]: Starting The Apache HTTP Server...
Feb 14 12:57:54 ip-172-31-14-195.ap-south-1.compute.internal systemd[1]: Started The Apache HTTP Server.
[root@ip-172-31-14-195 ec2-user]#
```

1. Give command
2. Systemctl enable httpd => to enable httpd service
3. Systemctl start httpd => to start httpd service
4. Systemstl status httpd => to check httpd service status
5. Curl localhost:80 => to check wheather apache server is responding on port 80

Steps to create EC2 Instance.



How to install Apache Web server on EC2 instance.

The screenshot shows a terminal window with the following session:

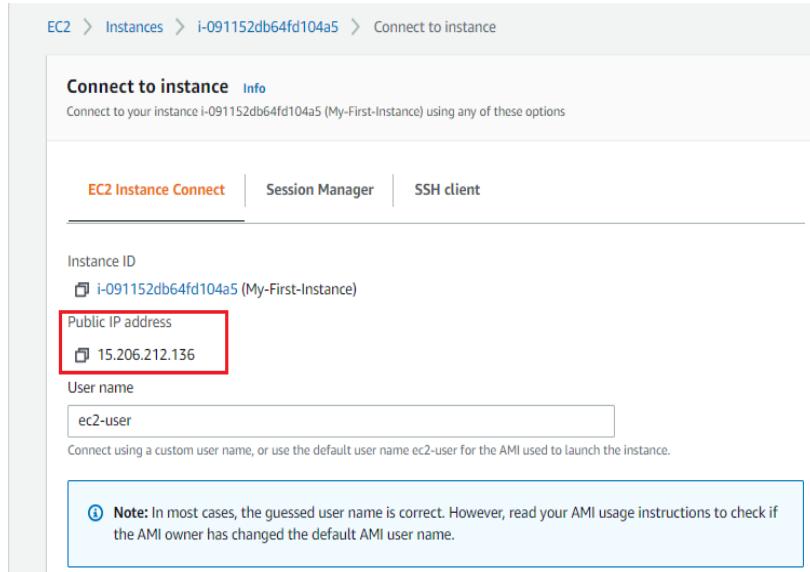
```
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]# hostname -f  
ip-172-31-14-195.ap-south-1.compute.internal  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]# echo "My Name is Bond. James Bond 007. $(hostname -f)" > /var/www/html/index.html  
[root@ip-172-31-14-195 ~]#  
[root@ip-172-31-14-195 ~]#
```

The command `echo "My Name is Bond. James Bond 007. $(hostname -f)" > /var/www/html/index.html` and its output are highlighted with a red box.

1. Give command
2. Hostname -f => to know name of your ec2 instance
3. Echo “My name is bond. James Bond 007. \$(hostname -f)” > /var/www/html/index.html

Steps to create EC2 Instance.

How to install Apache Web server on EC2 instance.



EC2 > Instances > i-091152db64fd104a5 > Connect to instance

Connect to instance Info

Connect to your instance i-091152db64fd104a5 (My-First-Instance) using any of these options

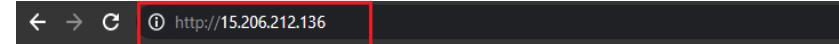
EC2 Instance Connect Session Manager SSH client

Instance ID: i-091152db64fd104a5 (My-First-Instance)

Public IP address: 15.206.212.136 (highlighted with a red box)

User name: ec2-user

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.



← → C ⓘ http://15.206.212.136



This site can't be reached

15.206.212.136 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT

1. Copy the public IP of ec2 instance and paste in browser.
2. You will get an error “ERR_CONNECTION_TIMED_OUT”

Lets Solve this issue now.....

Steps to create EC2 Instance.

How to solve connection error. Whenever there is timeout problem, check your “Security Groups”.



The screenshot shows the AWS Management Console interface for the EC2 service. On the left, a navigation menu lists various AWS services like Spot Requests, Savings Plans, Reserved Instances, etc. A red box highlights the 'Network & Security' section, specifically the 'Security Groups' link. The main content area displays a table of instances. One instance, 'My-First-Inst...', is selected and shown in detail below the table. The instance is running, of type t2.micro, and has 2/2 checks passed. It is located in the ap-south-1b availability zone and has a public IP address ec2-15-123-45-67. Below the table, there are sections for 'State transition message', 'Usage operation' (with a 'RunInstances' button), and 'RAM disk ID'.

Click on “Security Groups” from left side menu. Our EC2 instance is attached with “My-SG” security group. Select that and click on “Actions” then select “Edit Inbound Rules”.

The screenshot shows the AWS Management Console interface for the Security Groups service. The left sidebar includes the same navigation menu as the previous screenshot. The main table lists four security groups. The third group, 'My-SG', is selected and highlighted with a red box. The 'Actions' menu for this group is open, and the 'Edit inbound rules' option is also highlighted with a red box. Other actions listed in the menu include 'Edit outbound rules', 'Manage tags', 'Manage stale rules', and 'Copy to new security group'. The table columns include Name, Security group ID, Security group name, and VPC ID.

Steps to create EC2 Instance.

How to solve connection error. Whenever there is timeout problem, check your “Security Groups”.



The screenshot shows the AWS Security Groups Inbound rules configuration. It displays two rules:

- Rule 1:** Type: SSH, Protocol: TCP, Port range: 22, Source: Custom (0.0.0.0/0), Description: allow traffic from all IP addresses.
- Rule 2:** Type: HTTP, Protocol: TCP, Port range: 80, Source: Custom (0.0.0.0/0), Description: allow traffic from anywhere on port 80.

A red box highlights the "Add rule" button at the bottom left. Another red box highlights the second rule's source field. A note at the bottom states: "⚠ NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created." At the bottom right, there are "Cancel", "Preview changes", and "Save rules" buttons, with "Save rules" also highlighted by a red box.

Click on “Add Rule” . Select “HTTP”. And then click on “Save rules”.

Now, again go on browser and paste public IP and hit enter. You will get the output.

The screenshot shows a browser window with the address bar displaying "Not secure | http://15.206.212.136". The main content area shows the text "My Name is Bond. James Bond 007. ip-172-31-14-195.ap-south-1.compute.internal", which is highlighted by a red box.

Congratulations!!!!!!

Steps to create EC2 Instance.



Step 3: Clean up your instance

- 1) After you've finished with the instance that you created for this tutorial, you should clean up by terminating the instance.
- 2) If you launched an instance that is not within the [AWS Free Tier](#), you'll stop incurring charges for that instance as soon as the instance status changes to shutting down or terminated.

The screenshot shows the AWS EC2 Instances page. A single instance, "My-First-Inst...", is listed. A context menu is open over this instance, showing options like "Launch instances", "Launch instance from template", "Connect", and four options highlighted with red boxes: "Stop instance", "Start instance", "Reboot instance", and "Terminate instance". Another context menu is open over the "Instance state" column header, also showing "Stop instance", "Start instance", "Reboot instance", and "Terminate instance" options highlighted with red boxes. The instance details panel on the left shows the instance ID "i-091152db64fd1015", the instance type "t2.micro", and the owner information "Owner 933126026785".

Click on “Instance State” . Or right click on your “Instance Id”. Select “Stop instance” or “Terminate instance”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Steps to create EC2 Instance.



Step 3: Clean up your instance

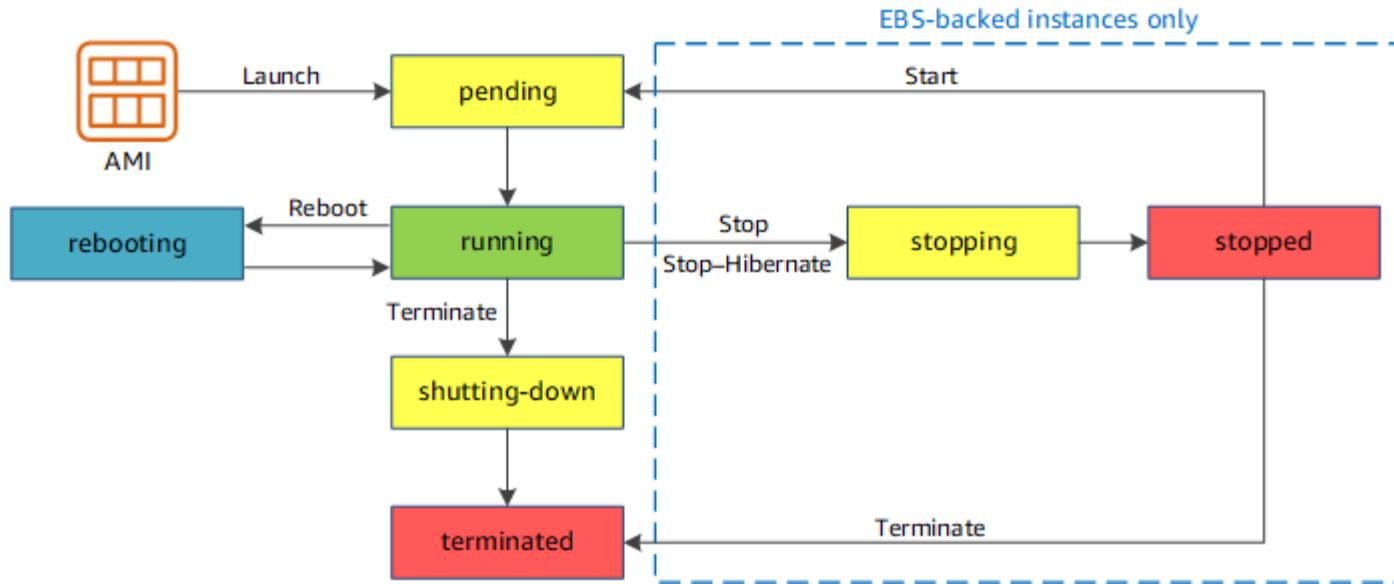
- 1) After you've finished with the instance that you created for this tutorial, you should clean up by terminating the instance.
- 2) If you launched an instance that is not within the [AWS Free Tier](#), you'll stop incurring charges for that instance as soon as the instance status changes to shutting down or terminated.

The screenshot shows the AWS EC2 Instances page. A single instance named "My-First-Inst..." is listed. The "Actions" menu for this instance is open, showing options: Stop instance, Start instance, Reboot instance, Hibernate instance, and Terminate instance. The "Terminate instance" option is highlighted with a red box. On the right side of the screen, the instance details panel shows the instance state as "t2.micro" and provides links for alarm status, availability zone, and public IP. At the bottom left, there's a "State transition message" field and an "Owner" section with a contact number.

Click on “Instance State” . Or right click on your “Instance Id”. Select “Stop instance” or “Terminate instance”.

Ref Link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Instance lifecycle



Ref Link: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html>

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- Compute services (EC2)
- **Storage services (S3 Buckets, EBS)**
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)

S3 (Simple Storage Service)



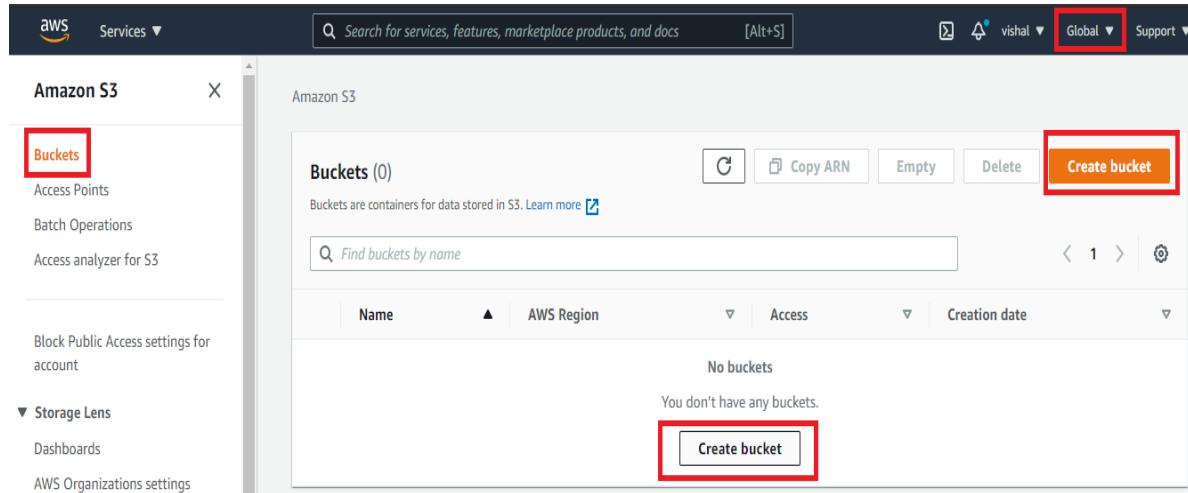
- ❖ Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- ❖ You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- ❖ Amazon S3 stores data as objects within buckets.
- ❖ An object is a file and any optional metadata that describes the file.
- ❖ To store a file in Amazon S3, you upload it to a bucket.
- ❖ When you upload a file as an object, you can set permissions on the object and any metadata.
- ❖ Buckets are containers for objects. You can have one or more buckets.
- ❖ You can control access for each bucket, deciding who can create, delete, and list objects in it.
- ❖ You can also choose the geographical Region where Amazon S3 will store the bucket and its contents and view access logs for the bucket and its objects.

Ref link: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

Use Case: Deploy your own static website on S3

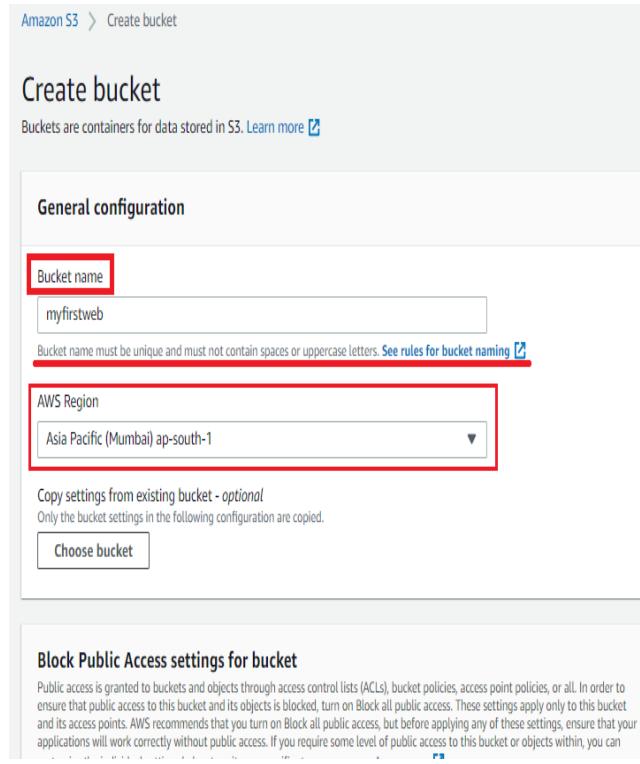
Steps:

- Step 1: Create a sample web application on your local machine.
- Step 2: Login to AWS console
- Step 3: Search for S3 service



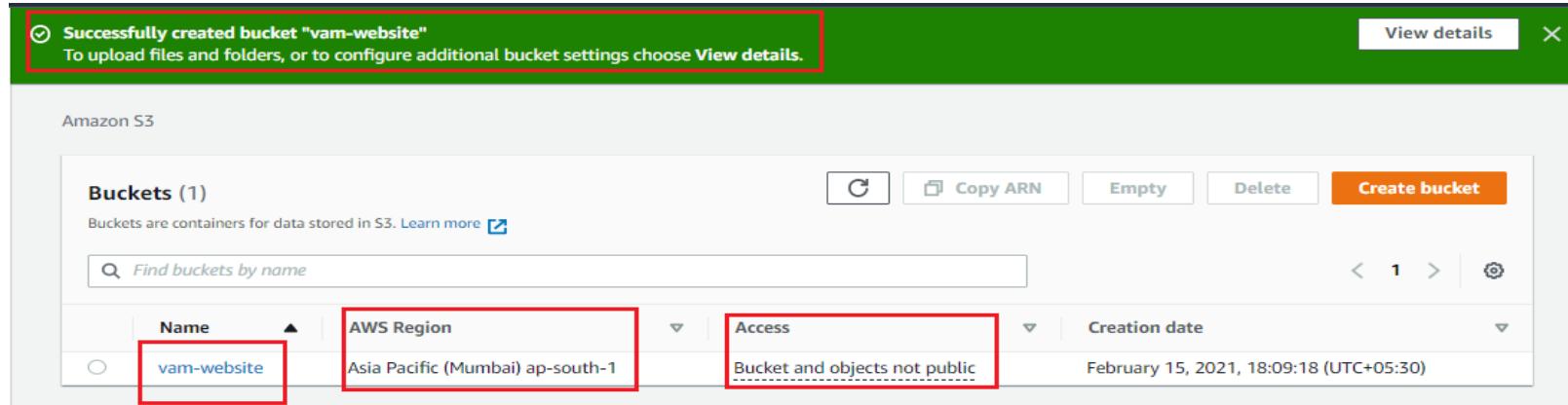
The screenshot shows the AWS S3 service page. On the left, there's a sidebar with options like 'Buckets' (which is selected and highlighted with a red box), 'Access Points', 'Batch Operations', 'Access analyzer for S3', 'Block Public Access settings for account', 'Storage Lens', 'Dashboards', and 'AWS Organizations settings'. The main area is titled 'Amazon S3' and shows a table with one row: 'Buckets (0)'. Below the table, it says 'No buckets' and 'You don't have any buckets.' At the bottom right of the table area, there's a large orange 'Create bucket' button, which is also highlighted with a red box. The top navigation bar includes the AWS logo, 'Services ▾', a search bar ('Search for services, features, marketplace products, and docs [Alt+S]'), and user information ('vishal ▾ Global ▾ Support ▾').

- Step 4: Click on “Create Bucket”
- Step 5: give “Bucket Name” which must be globally unique.
- Step 6: select region
- Step 7: Click on “Create Bucket”



The screenshot shows the 'Create bucket' wizard. The first step is 'General configuration'. It has a 'Bucket name' field containing 'myfirstweb' (which is highlighted with a red box) and an 'AWS Region' dropdown menu set to 'Asia Pacific (Mumbai) ap-south-1' (also highlighted with a red box). Below these fields, there's a note about copy settings from existing buckets and a 'Choose bucket' button. At the bottom, there's a section for 'Block Public Access settings for bucket' with a note about public access settings and a link to learn more.

Use Case: Deploy your own static website on S3



Successfully created bucket "vam-website"
To upload files and folders, or to configure additional bucket settings choose [View details](#).

View details X

Amazon S3

Buckets (1)

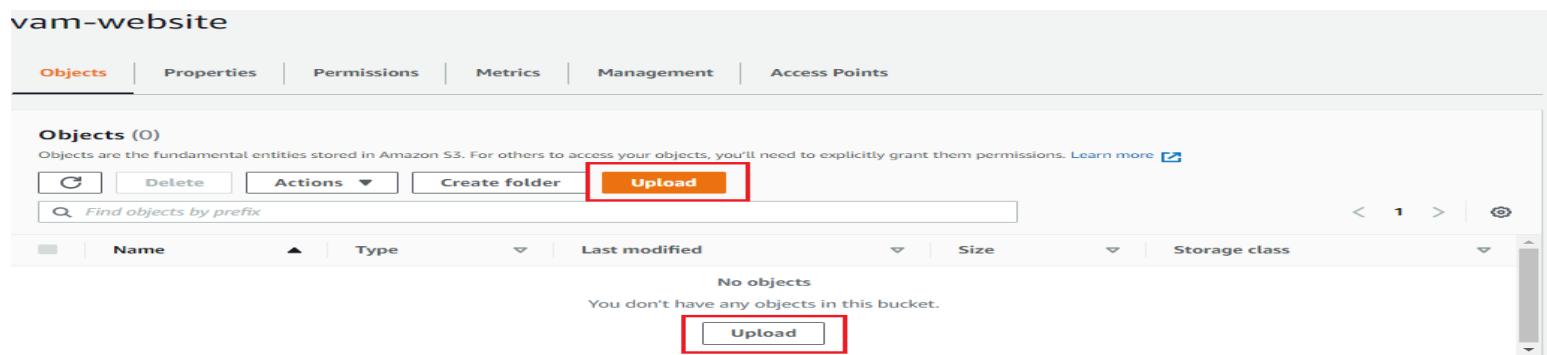
Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

Name	AWS Region	Access	Creation date
vam-website	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	February 15, 2021, 18:09:18 (UTC+05:30)

Congratulations!!!! - A bucket is successfully created.

Step 8: Right click on Bucket Name. and click on “Upload”



vam-website

Objects Properties Permissions Metrics Management Access Points

Objects (0)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions ▾ Create folder Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
No objects You don't have any objects in this bucket.				

Use Case: Deploy your own static website on S3

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.

Files and folders (0)					
All files and folders in this table will be uploaded.					
<input type="text"/> Find by name					
Name	Folder	Type	Size		
No files or folders					
You have not chosen any files or folders to upload.					

[Remove](#) [Add files](#) [Add folder](#)

Step 9: Click on “Add Folder”. Select your website folder and upload it. And click on “Upload” Button.

Step 10: go on your Object Page.

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.

Files and folders (3 Total, 862.0 B)					
All files and folders in this table will be uploaded.					
<input type="text"/> Find by name					
Name	Folder	Type	Size		
error.html	mywebsite/	text/html	269.0 B		
index.html	mywebsite/	text/html	312.0 B		
page2.html	mywebsite/	text/html	281.0 B		

[Remove](#) [Add files](#) [Add folder](#)

Destination

Destination
<s3://vam-website>

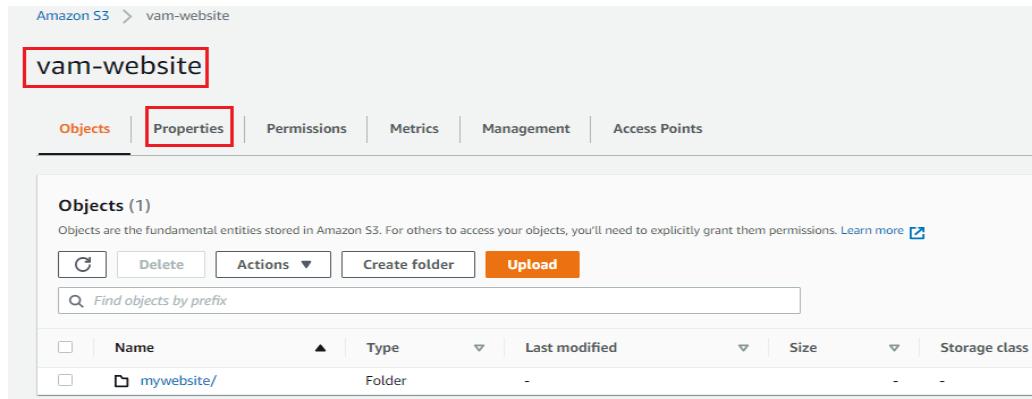
Use Case: Deploy your own static website on S3

Step 9: Click on “Add Folder”. Select your website folder and upload it. And click on “Upload” Button.

Step 10: go on your Object Page.

Step 11: Click on “Properties”

Step 12: At bottom “Static Website Hosting” click on “Edit”



Amazon S3 > vam-website

vam-website

Objects (1)

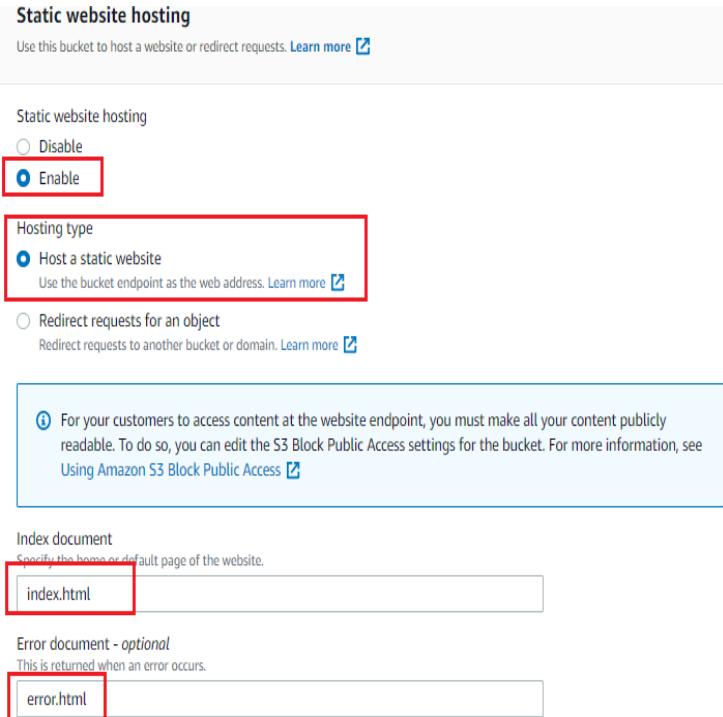
Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Delete

Name	Type	Last modified	Size	Storage class
mywebsite/	Folder	-	-	-

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)



Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Disable Enable

Hosting type

Host a static website

Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object

Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document

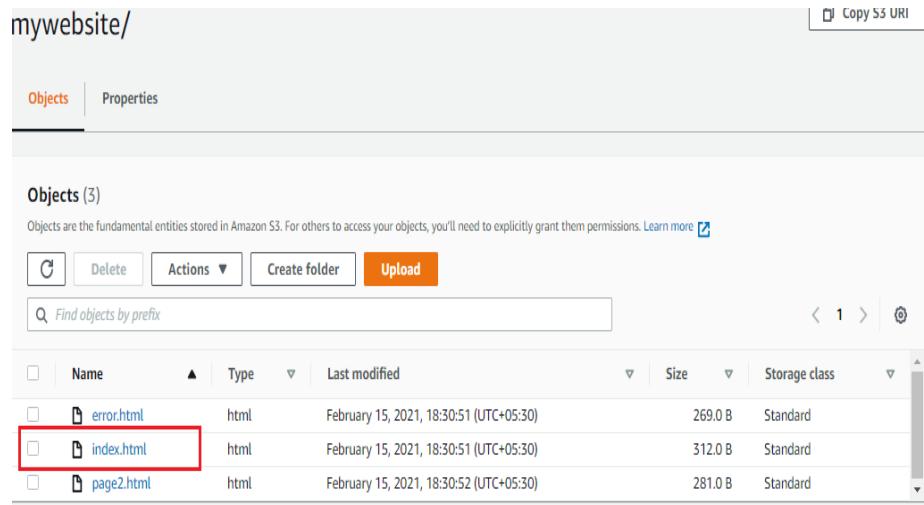
Specify the home or default page of the website.

Error document - optional

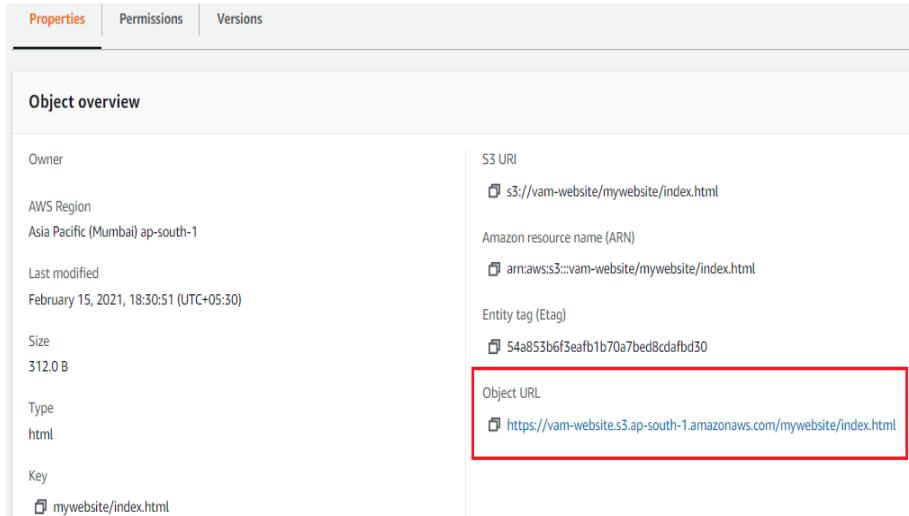
This is returned when an error occurs.

Use Case: Deploy your own static website on S3

Step 12: Click on “index.html” object. And get the object URL



The screenshot shows the AWS S3 console interface. At the top, there's a search bar with 'mywebsite/' and a 'Copy S3 URI' button. Below it, tabs for 'Objects' and 'Properties' are visible. Under 'Objects (3)', a table lists three files: 'error.html', 'index.html', and 'page2.html'. The 'index.html' row is selected and highlighted with a red box. The table columns include Name, Type, Last modified, Size, and Storage class. The 'index.html' file was last modified on February 15, 2021, at 18:30:51 (UTC+05:30) and has a size of 312.0 B.



The screenshot shows the 'Properties' tab for the 'index.html' object. It displays various details: Owner (AWS Region: Asia Pacific (Mumbai) ap-south-1), Last modified (February 15, 2021, 18:30:51 (UTC+05:30)), Size (312.0 B), Type (html), and Key (mywebsite/index.html). The 'Object URL' field, which contains the URL <https://vam-website.s3.ap-south-1.amazonaws.com/mywebsite/index.html>, is also highlighted with a red box.

Step 13: Copy the URL in browser and you will get an Error “Access Denied”.



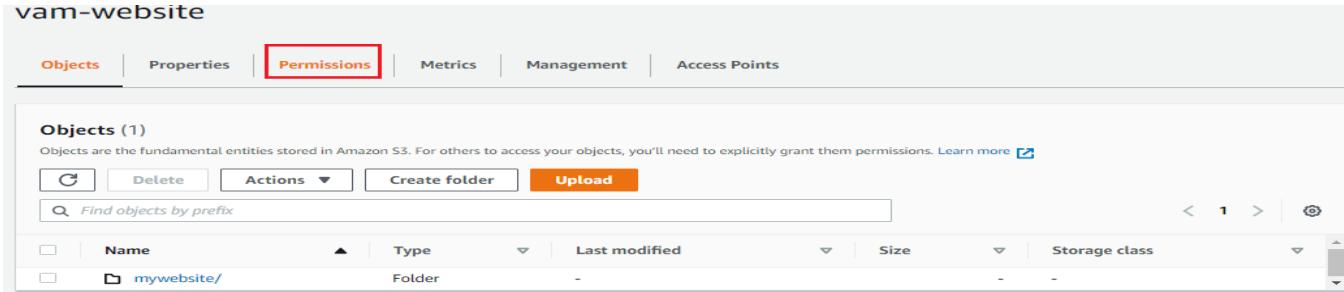
The screenshot shows a browser window with the URL <https://vam-website.s3.ap-south-1.amazonaws.com/mywebsite/index.html> in the address bar. The page content is an XML error document with the following text:

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.  
<Error>  
  <Code>AccessDenied</Code>  
  <Message>Access Denied</Message>  
  <RequestId>D9FF1B7487C054F0</RequestId>  
  <HostId>M5z/VKfygnJPVXlkqnqS2nXKdMY/RxpjnkZaF3eTja9p9COOXeF3hT5cDGo6Tf1Uj7xbxFWh4Q=</HostId>
```

Lets Solve this issue now.

Use Case: Deploy your own static website on S3

Step 14: Visit your bucket page and click on “Permission”



The screenshot shows the AWS S3 console for a bucket named 'vam-website'. The 'Permissions' tab is highlighted with a red border. Under the 'Objects' section, there is one object named 'mywebsite/'. The object type is listed as 'Folder'.

Step 15: Click on “Bucket Public access - Edit” . Uncheck the “Block all public access” box. Click on “Save Changes”

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

On

- Block public access to buckets and objects granted through new access control lists (ACLs)
 On
- Block public access to buckets and objects granted through any access control lists (ACLs)
 On
- Block public access to buckets and objects granted through new public bucket or access point policies
 On
- Block public and cross-account access to buckets and objects through any public bucket or access point policies
 On

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

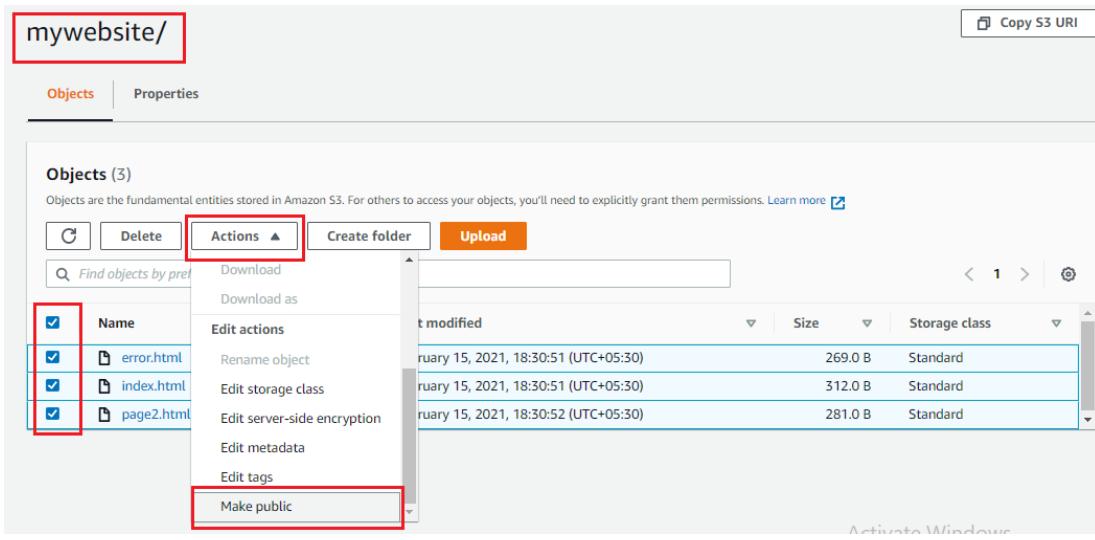
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Use Case: Deploy your own static website on S3

Step 16: again go to object page, copy “index.html” link and open in browser. If still problem remain then do the following steps:

Step 17: open Object page. Select all “Object” click on “Action” and select “Make Public”.



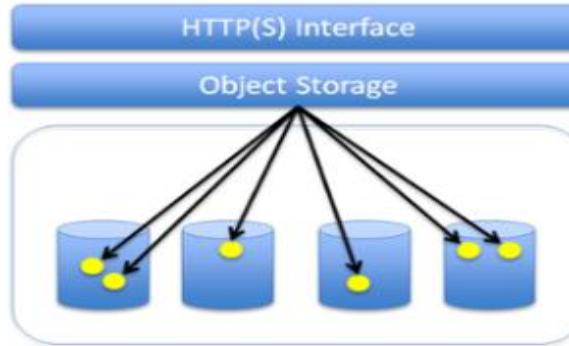
The screenshot shows the AWS S3 console with the path "mywebsite/" selected in the left sidebar. The main area displays three objects: "error.html", "index.html", and "page2.html". The "Actions" button is highlighted with a red box. A dropdown menu is open, showing options like "Edit actions", "Rename object", "Edit storage class", etc., followed by a "Make public" option at the bottom, which is also highlighted with a red box.



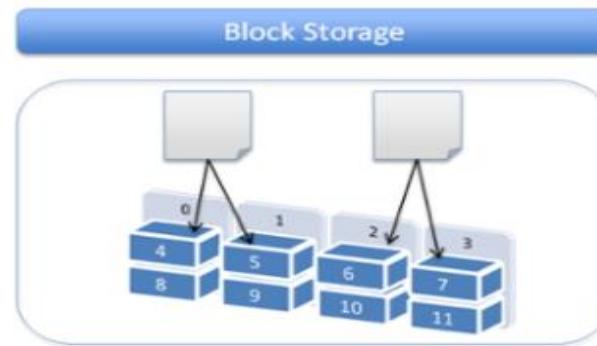
Step 18: Now copy the index.html url and open in browser.

Congratulations!!!!!!

Difference between S3, EBS and EFS



- Store virtually unlimited files.
- Maintain file revisions.
- HTTP(S) based interface.
- Files are distributed in different physical nodes.



- File is split and stored in fixed sized blocks.
- Capacity can be increased by adding more nodes.
- Suitable for applications which require high IOPS, database, transactional data.

Image Source: [NetApp Cloud](#) (used with permission)

Difference between S3, EBS and EFS

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	99.99%	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

Ref: <https://jayendrapatil.com/aws-s3-vs-ebs-vs-efs/>

Contents:

- What is AWS
- Why are enterprises choosing AWS?
- AWS Account setup and billing alarms
- IAM Introduction
- Compute services (EC2)
- Storage services (S3 Buckets, EBS)
- Database services (RDS, Dynamo DB)
- Elastic Load Balancer (ELB)
- Lambda

Thank You

What is Terraform & Infrastructure as Code (IaC)?

If you’re just getting started in software engineering, or you’ve been around a long time, you’ve probably at least heard the terms “Infrastructure as Code” and “Terraform” mentioned. But what are they, and why are they important?

Terraform is an Infrastructure as Code (IaC) tool that allows engineers to define their software infrastructure in code. While the idea of “code” may not be novel to engineers; the ability to provision infrastructure this way is a powerful abstraction that enables managing large distributed systems at scale.

In this article, we’ll take a look at what Infrastructure as Code and Terraform are, how they can help you in your work as a developer, and how you can get started using them.

Grab the Terraform cheat sheet

Check out the top 10 Terraform commands and get a full rundown of all the basic commands you need to get the most out of Terraform in our [Terraform cheat sheet](#).

What is Infrastructure as Code?

Infrastructure as Code is a way of defining and managing your infrastructure using code, rather than manual processes like clicking through a UI or using the command line. This means that you can manage your infrastructure in the same way that you manage your application code – with version control, automation, and collaboration. In other words, infrastructure as code is a way of making your infrastructure more like software.

Historically, Infrastructure as Code has seen many iterations, starting with configuration management tools like [CFEngine](#), [Chef](#), [Puppet](#), [Ansible](#), and [Salt](#). Newer tooling like [Cloudformation](#) and [Terraform](#) take a declarative approach and focus on the actual provisioning of resources, as opposed to the configuration of existing ones. The newest generation of tools focuses on using the capabilities of existing imperative programming languages. [AWS](#) and [Terraform](#) both provide Cloud Development Kits(CDKs), and [Pulumi](#) is also a popular option for provisioning infrastructure with traditional software tools.

What is Terraform?

Terraform is a tool for provisioning, managing, and deploying infrastructure resources. It is an open-source tool written in Golang and created by [the HashiCorp company](#). With Terraform, you can manage infrastructure for your applications across multiple cloud providers – AWS, Azure, GCP, etc. – using a single tool.

To get started with Terraform, developers simply need to download the Terraform binary, choose which [provider/platform](#) they’ll be working with, create some [boilerplate configuration for that provider](#), and they can get started creating infrastructure code.

One of the key features of Terraform is its declarative syntax. This means that you define what your desired end state is, and Terraform figures out the best way to achieve that. Compared to the imperative workflow of traditional programming languages, this can be a bit of a shift in mindset – but it enables managing infrastructure deployments at scale without the steep learning curve typical to software development.

The typical workflow for provisioning resources with Terraform is as follows:

1. Some Terraform configuration is written, including the provider definition.
2. The working directory is initialized(this is called the root module).
3. Provider plugins are downloaded.
4. The command `terraform plan` is run in the root module, generating a proposed plan to provision resources.
5. If the plan is acceptable, the `terraform apply` command is run and resources are provisioned.

Terraform keeps track of the state of the resources it manages in a [state file](#). This file is essentially a large JSON data structure that tracks proposed changes to infrastructure, as well as out-of-band changes that may have occurred to live resources outside of the Terraform configuration. New Terraform users typically maintain a local state file on their workstation or laptop. At scale with multiple engineers managing infrastructure, the state is typically broken down into multiple files and stored remotely using services like AWS S3.

Terraform is also “idempotent”, which means that repeated plan/apply cycles will not trigger a re-deploy of resources; only changes to the existing state will be reflected in a new plan and apply invocation.

Now that you’ve learned a little bit about what Terraform is and what it does, we can start to explore the “why” of Terraform and the benefits it provides.

What are the benefits of Terraform?

Utilizing Infrastructure as Code to manage and deploy infrastructure resources [unlocks several benefits for developers and engineers](#). Making Terraform your tool of choice for IaC confers additional benefits that allow developers to leverage easy-to-use tools to manage complex software architecture.

1. Terraform configuration is written using a declarative paradigm.

The best way to understand declarative code is to compare it to the more familiar pattern of imperative logic that most modern programming languages use.

Imagine a basic Python program that takes a series of numbers as input from the user, prints each number out in ascending order, then returns the sum of all the inputs. A programmer needs to write the code in a specific, logical order, or the program will fail. If the program attempts to sum all the numbers before the input is received, it is likely to generate some kind of exception or error. If the programmer wants the numbers to be sorted into the correct order, that will need to occur before they are printed as output.

In each part of the program, the programmer is having to specifically define both the logic and order of logic when the program executes. In contrast, declarative program syntax means the programmer needs to only “declare” the desired end state of the program. The compiler, in this case, the Terraform binary, is programmed to determine the best order of operations path through which they achieve the desired end state described in the configuration. This is especially helpful in dealing with cloud provider APIs, as many cloud resources have dependencies on the creation of other foundational resources before creation can proceed.

2. Hashicorp Configuration Language (HCL) is a Domain Specific Language (DSL).

Domain-specific languages are designed with a specific use case in mind, specialized to handle the requirements and constraints of a specific application or program domain.

If the utility of a DSL doesn’t seem immediately obvious, you should consider one of the most famous and widely used DSLs: HTML. HTML is a markup language that focuses on the domain of hyper-text (read: the internet). HTML does away with complex program logic and syntax and focuses on the specific use-case of content presentation.

In the case of Terraform, developers only need to learn a minimal amount of HCL syntax before they can be productive. As a DSL, Terraform makes development easier and more efficient by abstracting away the complexity inherent in general-purpose languages. That’s not to say that Terraform doesn’t have more complex logic; advanced users can use newer, imperative constructs like for-loops and if/then logic.

HCL is considered a superset of the [JSON language](#), which means it shares similar syntax, but also has additional features beyond the scope of JSON.

3. Terraform is widely adopted

Terraform is generally considered the industry standard when it comes to Infrastructure as Code tooling. It isn’t always good advice to go with the herd, but when it comes to technology implementation, choosing a tool with a large community, solid support base, and multi-year longevity is critically important. No one wants to have to explain to stakeholders and customers that the application is down because the code or platform is no longer supported!

The other benefit of wide adoption is the collective, shared knowledge of the community. Best practices are developed and shared, and an ecosystem of supporting tools and documentation can be built. A great example of the power of community support is the [awesome-terraform repository on Github](#), a curated list of tools, libraries, documentation, blogs, and more.

4. Terraform enables immutability

Immutable infrastructure makes managing complex distributed systems easier and safer and allows them to scale much more reliably. What exactly is meant by “immutable infrastructure”, and how does Terraform enable it?

Consider a hypothetical scenario: a developer needs to deploy some changes to fix a production application. They create their changes locally, run some tests and linting to validate the changes and check syntax, and now they’re ready to deploy. However, to get their code to run in the staging environment, they have to change some configuration to point to the staging database. Then they

need to give the QA team access to the staging servers. Everything manages to check out in the staging environment, but when deploying to production disaster strikes, the application stops serving traffic and an outage occurs.

Was the original code bad? Or was it the changes made in staging? Because the lines between environments and stages were blurred, it's nearly impossible to tell. Mutable infrastructure means changes can occur at any point in the lifecycle of an application or its infrastructure.

With immutable infrastructure, build, release, and deploy stages are kept separate. Once code changes are built, they are stamped with an immutable release tag. Further changes or fixes result in a new tag being generated. Developers and engineers know that a change that was made in a local development environment is the same change across different environments and deployment stages.

[The 12-factor app framework highlights this pattern in factor V.](#)

5. Terraform is modular

Modularity is an important feature in a variety of languages and systems. Abstracting logic and resources behind simple interfaces is one of the best ways to manage complexity at scale. As Terraform deployments grow more complex, developers can consider employing [modules](#) to encapsulate various resources in a reusable package.

A common use case is providing other development teams with Kubernetes clusters for testing and development. Normally, provisioning a Kubernetes cluster in a cloud provider like AWS requires a lot of boilerplate configuration and resources, even when using managed services. With modules, that configuration can be hidden behind a basic configuration interface. Developers can import the module, specify whatever inputs the module author has provided, and they can provision a complete stack without having to duplicate effort needlessly.

How can I use Terraform?

Although it's simple to get started, Terraform is a very powerful tool for provisioning infrastructure. The key for new users is to start small with basic configuration, and work towards fully automating their infrastructure as code.

New developers should start with something simple; a couple of basic resources without complex dependency chains. You don't need to start by trying to manage your entire production application environment on day 1: try using Terraform to manage the S3 bucket where deployment artifacts are stored, or Google DNS records for a frontend website.

Once you're comfortable, you can start to iterate and grow your Terraform usage. One of the first steps is to move from local state to a remote state file with locking. It's virtually impossible to safely manage larger Terraform deployments with multiple users without the state having a locking mechanism. When a Terraform plan or apply is running, locks prevent other users from making changes that could result in an inconsistent state or corruption.

With multiple users now contributing Terraform configuration, you can start to expand usage to cover more and more of your infrastructure resources, including networking, security, CDN, and more. As your infrastructure increases in complexity, consider encapsulating certain segments of your architecture in modules.

Finally, your team can start to use Terraform to provision resources as part of your CI/CD strategy. CI/CD is a complex topic that we won't cover here, but [GitLab](#) and [GitHub](#) both provide great batteries-included solutions for deployment automation that can be used with Terraform. Hashicorp [provides solid](#) documentation specifically targeted at users who are considering automating their Terraform deployments.

Conclusion

Why provision infrastructure with clicks and manual processes while writing application code? Infrastructure as Code tools like Terraform means that infrastructure configuration can be brought into the same development processes, allowing for testing, standardization, and scalability. The modern Infrastructure as Code ecosystem has a broad variety of resources for learning and getting started.

Want to learn more about Terraform?

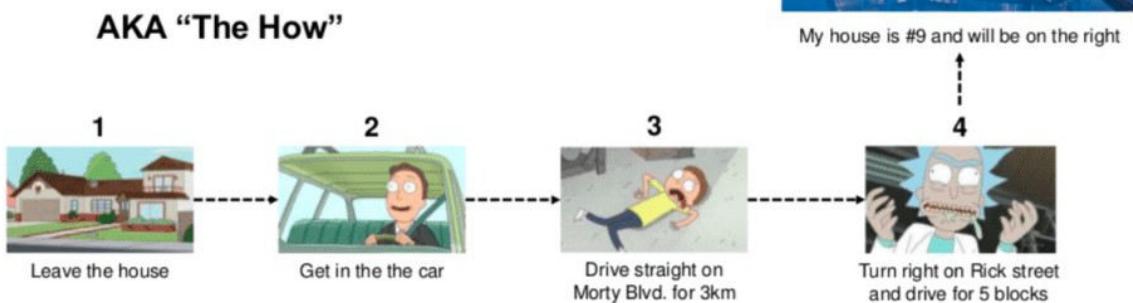
Why not try studying for the Hashicorp Certified: Terraform associate certification? Pluralsight offers an [excellent cert prep course](#) which teaches you everything you need to know about Terraform, including how to use it effectively in your own projects. It's a great way to learn about Terraform, even if you don't end up taking the certification exam.

Infrastructure as Code (IaC): Declarative vs Imperative

IaC: Imperative

Imperative (procedural):

Defines [specific commands](#) that need to be executed in the appropriate order to end with the desired conclusion.



Imperative – focus on the **actual provisioning process** and may reference a file containing a list of settings and configuration values.

Declarative – focused on the **desired end state** of deployment and relies on an interpretation engine to create and configure the actual resources.

When to use IaC

- You use a large amount of IaaS resources.
- Your infrastructure is rented from many different providers or platforms.

- You need to make regular adjustments to your infrastructure.
- You need proper documentation of changes made to your infrastructure.
- You want to optimize collaboration between administrators and developers.

Advantages of IaC

- Minimizing shadow IT inside of businesses and enabling quick, effective infrastructure upgrades made concurrently with application development
- Creating trackable and auditable configurations by enabling version-controlled infrastructure and configuration modifications.
- Maintaining infrastructure and settings in the appropriate condition while effectively managing configuration drift.
- Connecting directly to platforms for CI/CD.

Declarative (functional):
Defines the **desired state** and the system executes what needs to happen to achieve that desired state.

AKA “The What”



IaC Tools & Platforms – Overview

The slide is titled "DEVOPS IAC TOOLS OVERVIEW" in large, bold, white and blue text. The background is dark blue. At the bottom, there's a graphic illustrating various DevOps tools: Ansible (represented by a play button icon), Docker (represented by a yellow square icon), Chef (represented by a green icon with a chef hat), Vagrant (represented by a purple icon with a plane), and Puppet (represented by a yellow icon with a cube). Two stylized figures are standing in front of a large screen displaying a server stack diagram.

Terraform

The top infrastructure as code (IaC) solution for managing infrastructure across a variety of clouds, including AWS, Azure, GCP, Oracle Cloud, Alibaba Cloud, and even platforms like Kubernetes and Heroku, is [Terraform](#) by HashiCorp.



While assuring the intended state throughout the settings, Terraform may be utilized to facilitate any infrastructure provisioning and management use cases across many platforms and providers.

Ansible

[Ansible](#) is a free, open-source configuration management solution with IaC capabilities rather than a specialized infrastructure management tool. It is an agentless solution that works with both cloud and on-premises systems and may be used by SSH or WinRM.



It has limitations when it comes to maintaining that infrastructure but shines at provisioning infrastructure and configuration management.

Chef/Puppet

There are two effective configuration management tools: Chef and Puppet. Both seek to offer infrastructure management skills with configuration management and automation across the development process.

1. Chef was created with stronger collaboration features to be readily integrated into DevOps methods. It is used for configuration management and is Best used for Deploying and

configuring applications using a pull-based approach.



2. Puppet developed as a result of focusing on pure process automation. Currently, Puppet has automatic observers built in to detect configuration drift. It is a popular tool for configuration management and needs agents to be deployed on the target machines before



puppet can start managing them.

Tools Overview Table:

Tool	Tool Type	Infrastructure	Architecture	Approach	Manifest Written Language
puppet	Configuration Management	Mutable	Pull	Declarative	Domain Specific Language (DSL) & Embedded Ruby (ERB)
CHEF	Configuration Management	Mutable	Pull	Declarative & Imperative	Ruby
ANSIBLE	Configuration Management	Mutable	Push	Declarative & Imperative	YAML
SALTSTACK	Configuration Management	Mutable	Push & Pull	Declarative & Imperative	YAML
Terraform	Provisioning	Immutable	Push	Declarative	HashiCorp Configuration Language (HCL)

IaC DevOps Best Practices

Avoid automating everything right away

- Try not to automate everything right away if you are a company, an application, or a platform that is still in the early stages of development. This is due to the potential for rapid

change. You may start automating your platform's provisioning and maintenance after it has more or less reached a stable state.

Check and keep an eye on your setups

- Infrastructure as Code should and can still be tested since it is still code. Before deploying your servers, you should install testing and monitoring tools for IaC to look for flaws and inconsistencies.

The more rigid the better

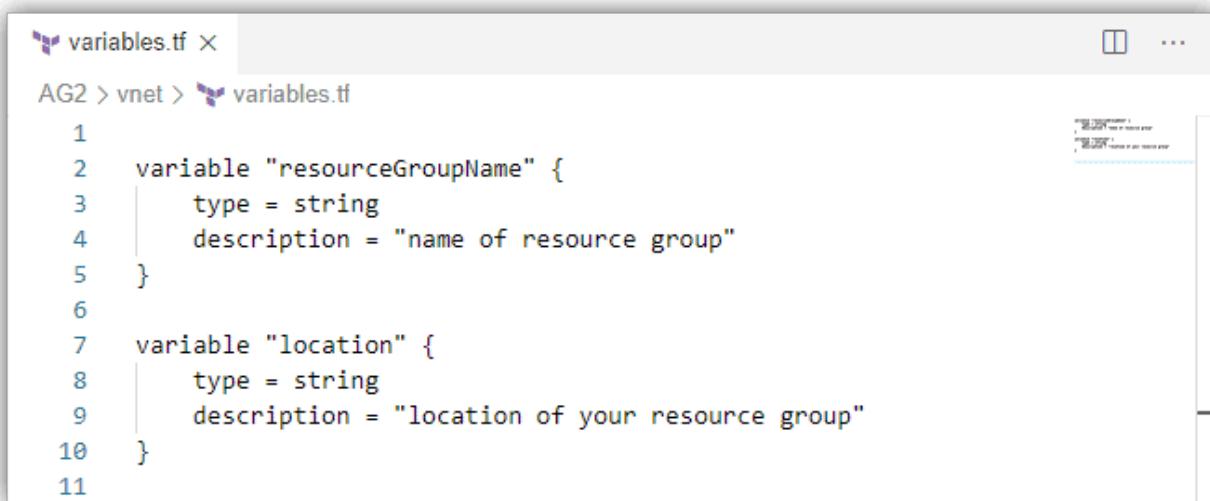
- Give as much detail as you can about the atmosphere you wish to create. Include the developers in the creation of the IaC standards for the runtime environments and infrastructure components. Making sure there are no flaws in the code can make it operate more efficiently.

Terraform Variables: Terraform Variable Types and Uses for Beginners

Terraform Input Variables

Terraform input variables are used as parameters to input values at run time to customize our deployments. Input terraform variables can be defined in the main.tf configuration file but it is a best practice to define them in a separate variable.tf file to provide better readability and organization.

A variable is defined by using a **variable** block with a label. The label is the name of the variable and must be unique among all the variables in the same configuration.



```
variables.tf
AG2 > vnet > variables.tf
1
2   variable "resourceGroupName" {
3     type = string
4     description = "name of resource group"
5   }
6
7   variable "location" {
8     type = string
9     description = "location of your resource group"
10  }
11
```

The variable declaration can optionally include three arguments:

- **description:** briefly explain the purpose of the variable and what kind of value is expected.
- **type:** specifies the type of value such as string, number, bool, map, list, etc.
- **default:** If present, the variable is considered to be optional and if no value is set, the default value is used.

Check out: How to [Install Terraform](#) in Linux, Mac, Windows

Terraform Input Variables

The type argument in a variable block allows you to enforce [type constraints](#) on the variables a user passes in. Terraform supports a number of types, including **string**, **number**, **bool**, **list**, **map**, **set**, **object**, **tuple**, and **any**.

If a type isn't specified, then Terraform assumes the type is any. Now, let's have a look at some of these terraform variables types and their classification.

Primitive Types

A *primitive* type is a simple type that isn't made from any other type. The available primitive types are:

- **string:** a sequence of characters representing some text, such as “hello”.
- **number:** a numeric value. The number type can represent both whole numbers like 15 and fractional values such as 6.28318.
- **bool:** either true or false.

Also Read: [Terraform Workflow](#).

Complex Types

A *complex* type is a type that groups multiple values into a single value. These values could be of a similar type or different types.

1. **List:** A Terraform list variable is a sequence of similar values indexed by numbers (starting with 0). It accepts any type of value as long as they are all of the same types. Lists can be defined either implicitly or explicitly.

```
# implicitly by using brackets [...]
variable "cidrs" { default = [] }

# explicitly
variable "cidrs" { type = "list" }
```

2. **Map:** A map is a collection of values where each value is identified by a string label. In the example below is a map variable named **managed_disk_type** to define the type of storage we want to use based on the region in Azure. In the default block, there are two string values, “Premium_LRS” and “Standard_LRS” with a named label to identify each one

westus2 and eastus.

```
variable "managed_disk_type" {
  type = map
  description = "Disk type Premium in Primary location Standard in DR location"

  default = {
    westus2 = "Premium_LRS"
    eastus = "Standard_LRS"
  }
}
```

3. **Object:** An object is a structural type that can contain different types of values, unlike map, list. It is a collection of named attributes that each have their own type.

In the below example, we have declared an object type variable **os** for the os image that can be used to deploy a VM.

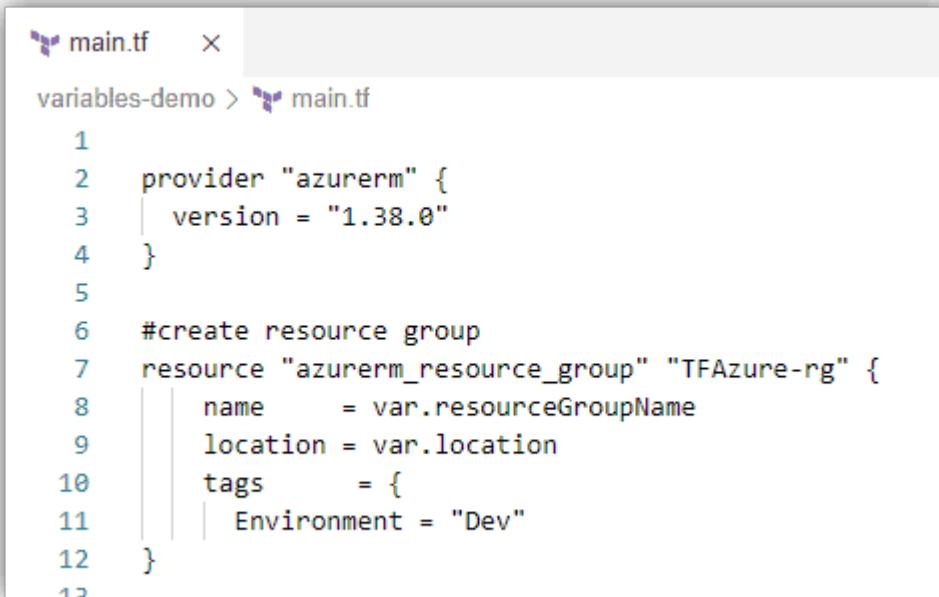
```
variable "os" {
  description = "OS image to deploy"
  type = object({
    publisher = string
    offer = string
    sku = string
    version = string
  })
}
```

Lists, maps, and objects are the three most common complex variable types. They all can be used for their specific use cases.

Also read: Step-by-step guide on [Terraform Certification](#)

Use Input Variables

After declaration, we can use the variables in our main.tf file by calling that variable in the **var.<name>** format. The value to the variables is assigned during **terraform apply**.



```
main.tf
variables-demo > main.tf
1
2 provider "azurerm" {
3   version = "1.38.0"
4 }
5
6 #create resource group
7 resource "azurerm_resource_group" "TFAzure-rg" {
8   name     = var.resourceGroupName
9   location = var.location
10  tags     = {
11    Environment = "Dev"
12  }
13 }
```

Also Check Our blog post on [Terraform Cheat Sheet](#). Click here

Assign Values To Input Variables

There are multiple ways to assign values to variables. The following is the descending order of precedence in which variables are considered.

1. Command-line flags

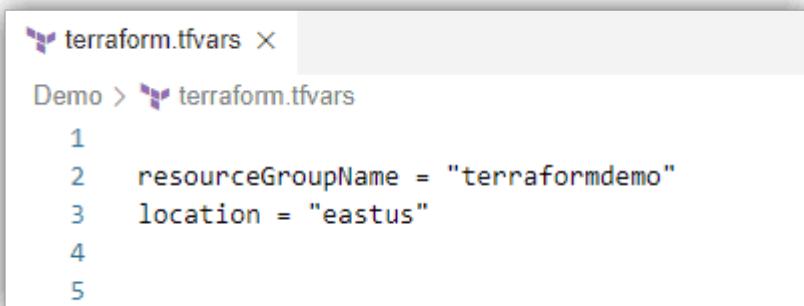
The most simple way to assign value to a variable is using the -var option in the command line when running the terraform plan and terraform apply commands.

```
$ terraform apply -var="resourceGroupName=terraformdemo-rg" -
var="location=eastus"
```

2. Variable Definition (.tfvars) Files

If there are many variable values to input, we can define them in a variable definition file. Terraform also automatically loads a number of variable definitions files if they are present:

- Files named exactly **terraform.tfvars** or **terraform.tfvars.json**
- Any files with names ending in **.auto.tfvars** or **.auto.tfvars.json**



```
terraform.tfvars
Demo > terraform.tfvars
1
2 resourceGroupName = "terraformdemo"
3 location = "eastus"
4
5
```

If the file is named something else, then use the **-var-file** flag directly to specify a file.

```
$ terraform apply -var-file="testing.tfvars"
```

3. Terraform Environment Variables

Terraform searches the environment of its own process for environment variables named **TF_VAR_<var-name>** followed by the name of a declared variable. Terraform scans all variables starting with TF_VAR and uses those as variable values for Terraform.

```
$ export TF_VAR_location=eastus
```

This can be useful when running Terraform in automation, or when running a sequence of Terraform commands in succession with the same terraform variables.

Read More: About [Hashicorp Terraform](#). Click here

Define Output Variables

Outputs allow us to define values in the configuration that we want to share with other resources or modules. For example, we could pass on the output information for the public IP address of a server to another process.

An output variable is defined by using an **output** block with a label. The label must be unique as it can be used to reference the output's value. Let's define an output to show us the public IP address of the server. Add this to any of the *.tf files.

```
output "public_ip_address" {
  description = "Public IP Address of Virtual Machine"
  value = azurerm_public_ip.publicip.ip_address
}
```

Multiple output

blocks can be defined to specify multiple output variables. Outputs are only shown when Terraform applies your plan, running a terraform plan will not render any outputs.

How to Provision AWS Infrastructure with Terraform?

Cloud itself is a big domain with various services running simultaneously. It will require a lot of effort for a person/organization to manage the cloud without automation. Thus, cloud automation is on its pace and various tools are proposed for faster and efficient development. One such automation tool is Terraform.

Prepare Your System

In this tutorial, we will use the Amazon EC2 instance with the Ubuntu system to run Terraform. But, it's your choice to run Terraform on any cloud platform or machine. For a better understanding, I will recommend you to follow the steps with us.

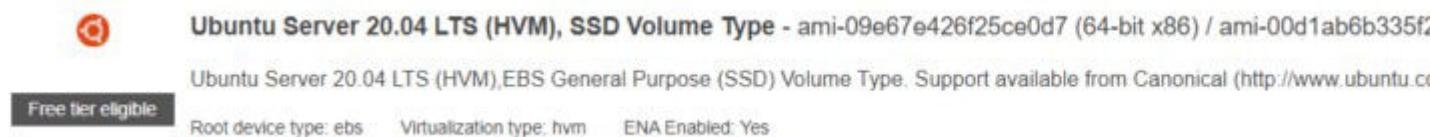
Step 1) Open up your [AWS console](#) or directly visit ‘console.aws.amazon.com‘. If you don’t have access to AWS, [create one free tier account](#).

Step 2) Search for EC2 in your AWS console and open it.

Step 3) Click on Launch Instances to create a new EC2 instance.



Step 4) Select an AMI (Amazon Machine Image). In our case, we will use Ubuntu.



Step 5) You can fill up all the details for your instance. To quickly create an EC2 instance, leave the settings to default, launch your instance and save your new RSA key pair safe.

	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	TerraformDemo	i-019be03da0a8d753c	Running	t2.micro

Step 6) After a few minutes, your instance will be live and running, as shown in the image above. Select it and click on connect to launch your EC2 instance.

Download, Install and Start Terraform

Based on your system requirement, you can download and install Terraform from the [official download page](#).

Step 1) From the official Terraform download page, copy the link to the Linux file as shown in the image below.



macOS
64-bit | Arm64



FreeBSD
32-bit | 64-bit | Arm



Linux
32-bit | 64-bit | Arm | Arm64



OpenBSD
32-bit | 64-bit



Solaris
64-bit



Windows
32-bit | 64-bit

Step 2) To download Terraform to your AWS instance, use ‘wget’ command followed by the copied download URL as shown in the image below.

```
wget  
https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
```

```
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-94-180:~$ wget https://releases.hashicorp.com/terraform/1.0.9/terraform_1.0.9_linux_amd64.zip
```

Step 3) To unzip the downloaded package, you need to install unzip tool in your system. Use the below command to download the tool.

```
sudo apt-get install unzip
```

Step 4) Check the downloaded package name using the ‘ls‘ command. Copy the package name and use the unzip command with the package name, as shown below.

```
ls  
unzip terraform_1.0.9_linux_amd64.zip
```

```
ubuntu@ip-172-31-94-180:~$ ls
terraform_1.0.9_linux_amd64.zip
ubuntu@ip-172-31-94-180:~$ unzip terraform_1.0.9_linux_amd64.zip
Archive:  terraform_1.0.9_linux_amd64.zip
  inflating: terraform
ubuntu@ip-172-31-94-180:~$
```

Step 5) Now, the Terraform is extracted successfully. Run the below commands one by one to run the terraform commands hassle-free by ignoring the directories. Finally, use the command ‘`terraform`’ to activate terraform, as shown in the image below.

```
echo $"export PATH=\$PATH:$pwd)" >> ~/.bash_profile
source ~/.bash_profile

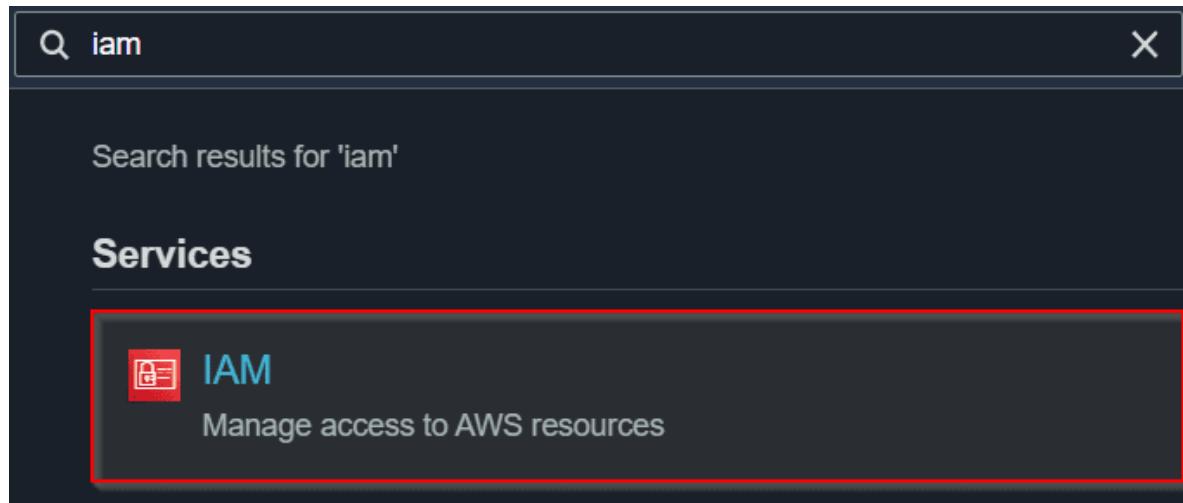
terraform
```

```
ubuntu@ip-172-31-94-180:~$ echo $"export PATH=\$PATH:$pwd)" >> ~/.bash_profile
ubuntu@ip-172-31-94-180:~$
ubuntu@ip-172-31-94-180:~$ source ~/.bash_profile
ubuntu@ip-172-31-94-180:~$
ubuntu@ip-172-31-94-180:~$ terraform
```

Generate AWS IAM Access Key

We installed terraform in our system, but now the question arises: How Terraform will provision anything without AWS permission. For this, we create an access key that will be used for resource provisioning by Terraform.

Step 1) Search and visit IAM in your AWS console.



Step 2) Select *Users* and *Add Users* here if you don't have one, as shown in the image below.

The screenshot shows the AWS IAM 'Users' page. On the left sidebar, under 'Access management', the 'Users' option is selected and highlighted with a red box. At the top right, there are 'Delete' and 'Add users' buttons; the 'Add users' button is also highlighted with a red box. The main area displays a table with columns for 'User name', 'Groups', 'Last activity', 'MFA', 'Password age', and 'Active key age'. A search bar at the top says 'Q, Find users by username or access key'. Below the table, it says 'No resources to display'.

Step 3) Name the user and select Access Key in the checkbox below. Now go to the Next window that is permission settings.

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

The form has a 'User name*' field containing 'terraformdemo', which is highlighted with a red box. Below it is a blue '+ Add another user' button.

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type*** **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Step 4) Here, create one new user group.



Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

[Create group](#)

Step 5) Name the group and search for required permission for allowing Terraform to perform its task. In our case, we will be creating one EC2 instance, so we selected *AmazonEC2FullAccess* permission, as shown in the image below. Now to generate this user group, click on the button Create.

Policy name	Type	Used as	Description
AmazonEC2FullAccess	AWS managed	Permissions policy (1), Boundary (1)	Provides full access to Amazon EC2 via the AWS Management Console.

Step 6) Select the created group for the IAM user and proceed with the final steps to make the new user active.

Group	Attached policies
terraformdemo	AmazonEC2FullAccess

Step 7) After creating the user, you will see the keys on your screen or visit the user to find them. Copy your Access Key ID and Secret Access Key as they will be used while provisioning AWS resources using Terraform.

User	Access key ID	Secret access key
terraformdemo	AKIAVDDOMWIVKZADFPXXNK	ofiE6ansggIiXh0II3qHkwJfXz8GgmgQBfl5CNL Hide

Create EC2 Instance with Terraform

For an easy understanding, we will create one EC2 instance using the terraform file with all the instructions to create the EC2 instance.

Step 1) Create a new directory using the ‘mkdir’ command and name it whatever you want. Then, visit the directory using the below commands.

```
mkdir terraform-lab
```

```
cd terraform-lab/
```

Step 2) Create a new file here that will have the instruction to provision the AWS resource. Use the below commands to create the file.

```
vim ec2.tf
```

Step 3) After running the above command, the file is created but to enable the edit mode, you need to press the INSERT or any other button on your keyboard. Copy the below code, replace it with your IAM keys and paste it on your file. Also, don't forget to maintain proper spacing and lines in your code, as shown in the image below.

The provider section in code uses the credentials but if you have an AWS CLI setup you may not require this. The resource section in code will create the resource *aws-instance* with the name *example* with mentioned *AMI (Amazon Machine Image)*.

```
provider "aws" {
  access_key = "ACCESS_KEY_HERE"
  secret_key = "SECRET_KEY_HERE"
  region     = "us-east-1"
}

resource "aws_instance" "example" {
  ami          = "ami-2757f631"
  instance_type = "t2.micro"
}
```

```
provider "aws" {
  access_key = "AKIAVDDGJWJZLAFPXXNK"
  secret_key = "ofiE6ansgqljXh1U2gtkxwJfxz0DqngQBfI5CNL"
  region     = "us-east-1"
}

resource "aws_instance" "example" {
  ami = "ami-2757f631"
  instance_type = "t2.micro"
}
```

Step 4) For saving the above file in the Ubuntu system, press the ‘*Esc*‘ button on your keyboard, type ‘:wq’ and press ‘*Enter*‘.

Step 5) Now everything is ready, use the below command to initialize terraform to compile the file. You will receive an error here if there is any syntax error in the file.

```
terraform init
```

```
ubuntu@ip-172-31-94-180:~/terraform-lab$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.63.0...
- Installed hashicorp/aws v3.63.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ip-172-31-94-180:~/terraform-lab$ █
```

Step 6) Finally, proceed with resource provisioning with the below command. You will receive an error here if the access keys are wrong, AMI (Amazon Machine Image) is wrong, or permission to provision EC2 Instance is not provided in the IAM (Identity Access Management).

If everything goes well, you will see a confirmation on your screen to provision. Type yes to proceed, and your resource will be live in a few minutes.

```
terraform apply
```

```
Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.
```

```
Enter a value: yes
```

```
aws_instance.example: Creating...  
aws_instance.example: Still creating... [10s elapsed]  
aws_instance.example: Still creating... [20s elapsed]  
aws_instance.example: Still creating... [30s elapsed]  
aws_instance.example: Still creating... [40s elapsed]  
aws_instance.example: Still creating... [50s elapsed]  
aws_instance.example: Still creating... [1m0s elapsed]  
aws_instance.example: Still creating... [1m10s elapsed]  
aws_instance.example: Still creating... [1m20s elapsed]  
aws_instance.example: Creation complete after 1m22s [id=i-0c56b6d7]
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
ubuntu@ip-172-31-94-180:~/terraform-lab$ █
```

Step 7) You can check your new AWS EC2 instance created in your AWS console. To delete all the resources created by Terraform, use the below command to clear all the things.

```
terraform destroy
```

Conclusion

Terraform is one of the most used Infrastructures as a Code tool by IT companies. In this blog, we created an AWS EC2 instance with Terraform using declarative syntax on our system. Using similar steps, we can create, manage and destroy any resource in AWS Infrastructure with the help of Terraform.

Terraform vs. Ansible: Key Differences and Comparison of Tools

Ansible vs Terraform battle continues to escalate as the DevOps environment focuses more on automation and orchestration. These two tools help in automating configurations and deploying infrastructure. Terraform offers to deploy Infrastructure as a Code, helps in readability and lift and shift deployments. Ansible is a configuration management tool for automating system configuration and management.

Terraform vs. Ansible

Terraform is a tool designed to help with the provisioning and deprovisioning of cloud infrastructure using an infrastructure as code approach. It is highly specialized for this purpose. On the other hand, Ansible is a more general tool that can be used for automation across various

domains. Both Terraform and ansible have strong open-source communities and commercial products that are well supported.

To explain in short, **Terraform is a tool used for creating and managing IT infrastructure. Ansible automates provisioning, deployment, and other IT processes.**

Similarities between Terraform and Ansible

Given the features of both technologies, Terraform and Ansible appear to be extremely similar tools at a high level.

- They are both capable of setting up the new cloud infrastructure and equipping it with the necessary application components.
- On the freshly formed virtual machine, remote commands can be carried out by both Terraform and Ansible. This indicates that neither tool requires an agent. Agent deployment on the computers is not necessary for operational reasons.
- Terraform builds infrastructure utilizing the APIs of cloud providers, and SSH is used for simple configuration operations. The same is true of Ansible; all necessary configuration activities are carried out over SSH. Both tools are masterless since the “state” information for neither requires a separate piece of infrastructure to manage.

Difference between Terraform and Ansible Provisioning (Terraform vs. Ansible)

Let's see how Terraform vs. Ansible battle differentiates from each other:

Terraform

Terraform is a provisioning tool.

It follows a declarative Infrastructure as a Code approach.
It is the best fit for orchestrating cloud services and setup
cloud infrastructure from scratch.

Terraform does not support bare metal provisioning by
default.

It does not provide better support in terms of packaging and
templating.

It highly depends on lifecycle or state management.

Ansible

Ansible is a configuration management tool.

It follows both declarative & procedural approaches.
It is mainly used for configuring servers with the
required software and updating already configured resources.

Ansible supports the provisioning of bare metal servers.

It provides full support for packaging and templating.

It does not have lifecycle management at all.

1. Orchestration vs. Configuration Management

Terraform and Ansible have so many similarities and differences at the same time. The difference comes when we look at two significant concepts of DevOps: Orchestration and configuration management.

Configuration management tools solve the issues locally rather than replacing the system entirely. Ansible helps to configure each action and instrument and ensures smooth functioning without any damage or error. In addition, Ansible comes up with hybrid capabilities to perform both orchestration and replace infrastructure.

Orchestration tools ensure that an environment is in its desired state continuously. Terraform is explicitly designed to store the state of the domain. Whenever there is any glitch in the system, terraform automatically restores and computes the entire process in the system after reloading. It is the best fit in situations where a constant and invariable state is needed. **Terraform Apply** helps to resolve all anomalies effectively.

Let's have a look at the Procedural and Declarative nature of Terraform and Ansible.

2. Declarative vs. Procedural

There are two main categories of DevOps tools: Procedural vs. Declarative. These two categories tell the action of tools.

Terraform follows the **declarative approach**, ensuring that if your defined environment suffers changes, it rectifies those changes. This tool attempts to reach the desired end state described by the *sysadmin*. Puppet also follows the declarative approach. With terraform, we can automatically describe the desired state and figure out how to move from one state to the next.

Ansible is of hybrid nature. It follows both declarative and **procedural style** configuration. It performs ad-hoc commands to implement procedural-style configurations. Please read the [documentation](#) of Ansible very carefully to get in-depth knowledge of its behavior. It's important to know whether you need to add or subtract resources to get the desired result or need to indicate the resources required explicitly.

3. Mutable vs. Immutable

A workflow for application deployment involves providing the infrastructure, installing the correct version of the source code, and installing any dependencies.

The infrastructure that serves as the foundation for later versions of apps and services has a property known as mutability. Either existing infrastructure is used for deployment, or we can create an entirely new set of infrastructure for it.

It depends on the deployment procedures whether the infrastructure is mutable or immutable. It is said to as malleable when subsequent versions of apps are released on the same infrastructure. However, it is said to as immutable if the deployment takes place during releases on entirely new infrastructure.

Although mutability appears convenient, there is a higher chance of failure. The previous version must first be uninstalled before the desired version can be installed when application configurations are applied again on the same infrastructure. Additional steps increase the likelihood of failure. This can lead to inconsistent setups and unpredictable behavior across a fleet of servers.

Instead, if we concentrate on minimizing these steps by skipping the uninstalling process and carrying out the installation on fresh infrastructure resources, we will have the opportunity to test the new deployment and roll it back in case it doesn't work. This approach to treating infrastructure as immutable gives administrators more control over making changes.

4. State Management

The full lifecycle of the resources under Terraform's administration is managed. It keeps up the state files' mapping of infrastructure resources to the most recent configuration. State management is crucial to Terraform's operation.

States are used to monitor configuration changes and provide the same. Additionally, it is possible to import preexisting resources managed by Terraform by bringing in state files from the infrastructure of the real world.

It is possible to query the Terraform state files at any moment to learn about the infrastructure components and their available characteristics.

Ansible, in contrast, does not provide any form of lifecycle management. Any changes made to the configuration are automatically implemented on the target resource because Ansible focuses on configuration management and assumes immutable infrastructure by default.

Terraform vs Ansible Provisioning



Terraform deals with **infrastructure automation**. Its current declarative model lacks some features which arise complexity. Using Terraform, the elements of required environments are separately described, including their relationships. It assesses the model, creates a plan based on dependencies, and gives optimized commands to Infrastructure as a Service. If there is no change in the environment or strategy, repeated runs will do nothing. If there is any **update** in the plan or environment, it will **synchronize** the cloud infrastructure.

Ansible follows a **procedural approach**. Various users create playbooks that are evaluated through top to bottom approach and executed in sequence. **Playbooks** are responsible for the configuration of network devices that contributes towards a procedural approach. Of course, Ansible provisions the cloud infrastructure as well. But its procedural approach limits it to large infrastructure deployments.

How does Terraform work?

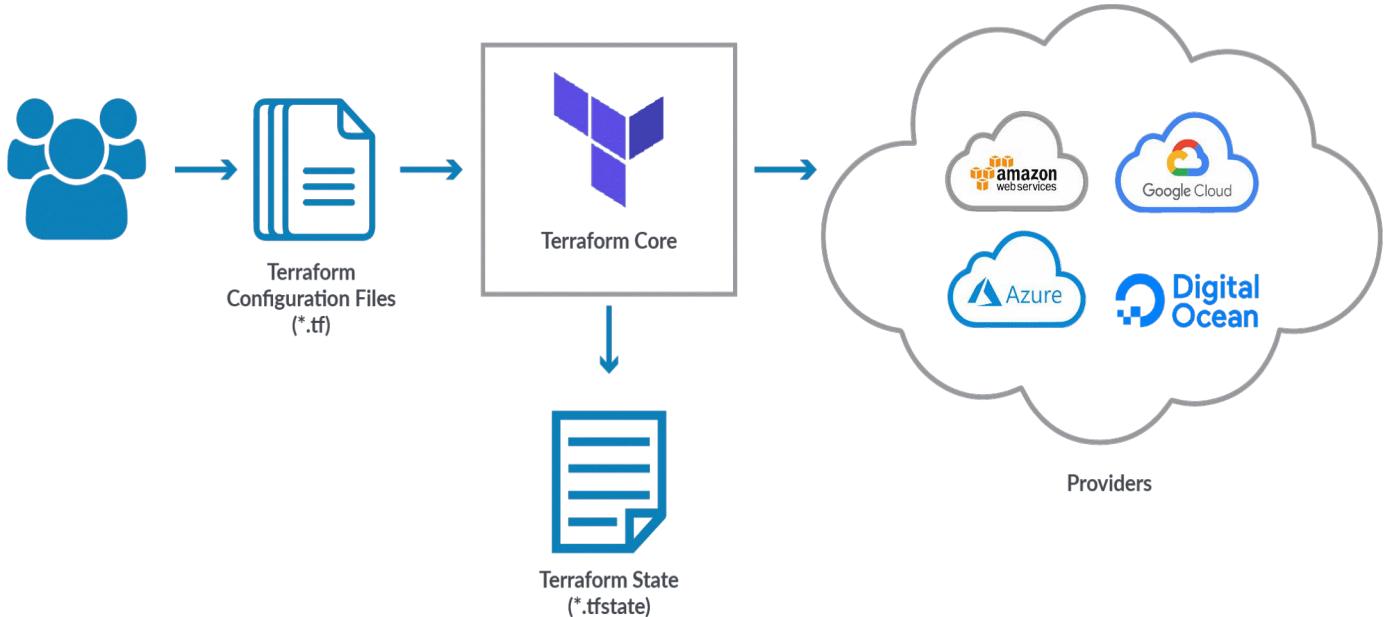
There are two main working components of terraform.

- **Terraform Core**
- **Providers**

Terraform is of **declarative nature**. It directly describes the end state of the system without defining the steps to reach there. It works at a high level of abstraction to describe what services and resources should be created and defined.

Terraform core takes two input sources to do its job. The first input source is a **terraform configuration** that is configured by its users. Users define what needs to be provisioned and created. The second input source is a state that holds information about the infrastructure.

So terraform core takes the input and figures out various plans for what steps to follow to get the desired output.



The second principal component is **providers**, such as cloud providers like [AWS](#), [GCP](#), [Azure](#), or other Infrastructure as service platforms. It helps to create infrastructure on different levels. Let's take an example where users create an AWS infrastructure, deploy Kubernetes on top of it, and then create services inside the cluster of Kubernetes. Terraform has multiple providers for various technologies; users can access resources from these providers through terraform. This is the basic working terminology of terraform that helps to provision and cover the complete application set up from infrastructure to fully developed application.

Check out: [Terraform cheat sheet](#).

Features of Terraform

As we have discussed the working of Terraform, now we will look at the features of Terraform.

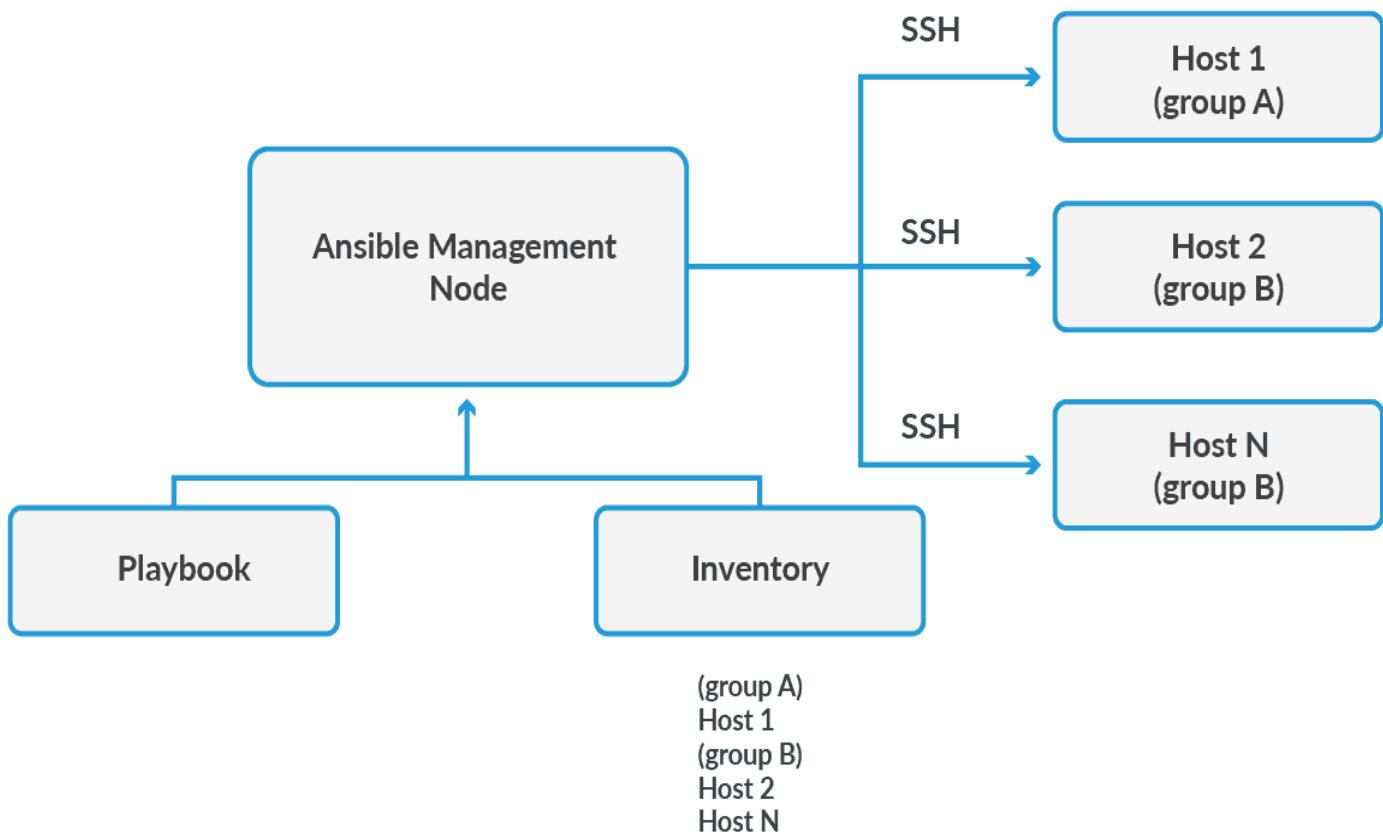
- Terraform follows a **declarative approach** which makes deployments fast and easy.
- It is a convenient tool to display the resulting model in a **graphical form**.
- Terraform also manages **external service providers** such as cloud networks and in-house solutions.
- It is one of the rare tools to offer **building infrastructure** from scratch, whether public, private or multi-cloud.
- It helps **manage parallel environments**, making it a good choice for testing, validating bug fixes, and formal acceptance.

- Modular code helps in achieving **consistency, reusability, and collaboration**.
- Terraform can **manage multiple clouds** to increase fault tolerance.

Also Check: Our blog post on [Terraform Workflow](#). Click here

How does Ansible work?

Ansible is agentless and doesn't run on target nodes. It makes **connections using SSH** or other authentication methods. It installs various **Python modules** on the target using JSON. These modules are simple instructions that run on the target. These modules are executed and removed once their job is done. This strategy ensures that there is no misuse of resources on target. Python is mandatory to be installed on both the controlling and the target nodes.



Ansible **management node** acts as a controlling node that controls the entire execution of the playbook. This node is the place to run the installations. There is an **inventory file** that provides the host list where the modules need to be run. The management node makes SSH connections to execute the modules on the host machine and installs the product. Modules are removed once they are installed in the system. This is the simple working process of Ansible.

Let's have a look at the features of ansible.

Features of Ansible

Now we will discuss various features Ansible provides to benefit its users.

- Ansible is used for **configuration management** and follows a procedural approach.
- Ansible deals with **infrastructure platforms** such as bare metal, cloud networks, and virtualized devices like hypervisors.

- Ansible follows **idempotent behavior** that makes it to place node in the same state every time.
- It uses **Infrastructure as a Code system configuration** across the infrastructure.
- It offers **rapid and easy deployment** of multi-tier apps with being agentless.
- If the code is **interrupted**, it allows entering the **code again** without any conflicts with other invocations.

Check out: [Ansible Configuration Management Tools](#)

Which one to choose: Terraform or Ansible?



Terraform vs. Ansible: Every tool has its unique characteristics and limitations. Let's check out which one to go with.

Terraform comes with good **scheduling capabilities** and is **very user-friendly**. It integrates with docker well, as docker handles the configuration management slightly better than Terraform. But there is no clear evidence of how the target devices are brought to their final state, and sometimes, the final configuration is unnecessary.

Ansible comes with **better security** and **ACL functionality**. It is considered a mature tool because it adjusts comfortably with **traditional automation frameworks**. It offers simple operations and helps to code quickly. But, on the other hand, it is not good at services like logical dependencies, orchestration services, and interconnected applications.

You can now choose between these two, according to the requirement of the situation and the job. For example, if the containerized solution is used to provision software within the cloud, then Terraform is preferable. On the other hand, if you want to gain reasonable control of your devices and find other ways to deploy underlying services, Ansible is more suitable. These tools will provide more comprehensive solutions in the future.

Conclusion

It is essential to know which tool is used for which job among Terraform vs. Ansible. Terraform is mainly known for provisioning infrastructure across various clouds. It supports more than 200 providers and a great tool to manage cloud services below the server. In comparison, Ansible is optimized to perform both provisioning and configuration management. Therefore, we can say that

both Terraform and Ansible can work hand in hand as **standalone tools** or **work together** but always pick up the right tool as per the job requirement.

Terraform Providers Overview

Terraform Providers: Terraform is one of the most popular tools used by DevOps teams to automate infrastructure tasks. It is used to provision and manage any cloud, infrastructure, or service.

Terraform officially supports around 130 providers. Its community-supported providers' page lists another 160. Some of those providers expose just a few resources, but others, such as AWS, OCI, or Azure, have hundreds of them.

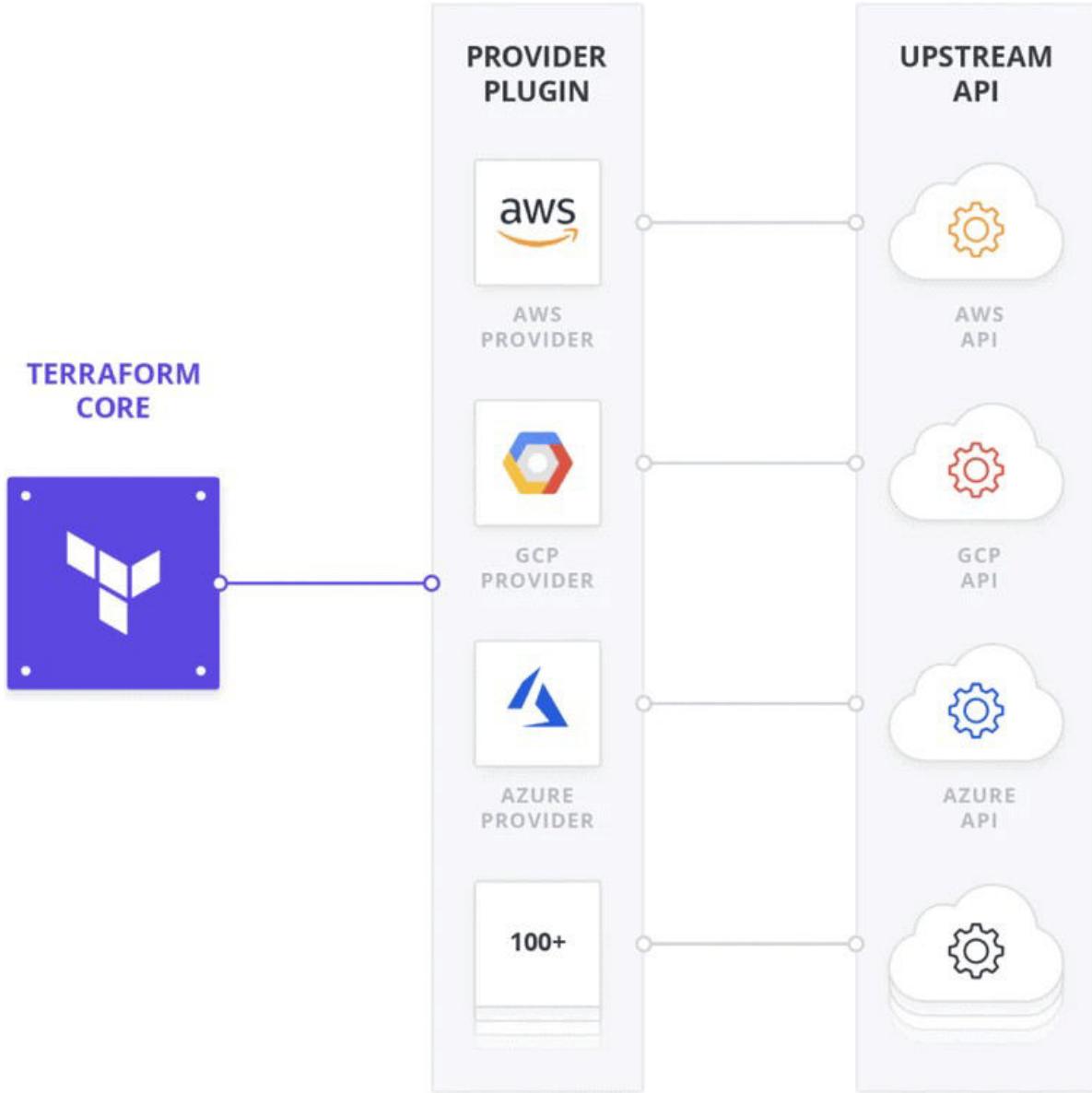
In this blog post, we cover a basic introduction of terraform providers and some major terraform cloud providers such as AWS, Azure, Google, and OCI.

A large percentage of Terraform users provision their infrastructure on the major cloud providers such as AWS, Azure, OCI, and others. To know more about various other terraform providers check [here](#).

Let's understand the basic terminologies often used in Terraform:

- **Terraform** provisions, updates, and destroys infrastructure resources such as physical machines, VMs, network switches, containers, and more.
- **Configurations** are code written for Terraform, using the human-readable HashiCorp Configuration Language (HCL) to describe the desired state of infrastructure resources.
- **Providers** are the plugins that Terraform uses to manage those resources. Every supported service or infrastructure platform has a provider that defines which resources are available and performs API calls to manage those resources.
- **Modules** are reusable Terraform configurations that can be called and configured by other configurations. Most modules manage a few closely related resources from a single provider.
- **The Terraform Registry** makes it easy to use any provider or module. To use a provider or module from this registry, just add it to your configuration; when you run `terraform init`, Terraform will automatically download everything it needs.

A provider is responsible for understanding API interactions and exposing resources. It interacts with the various APIs required to create, update, and delete various resources. Terraform configurations must declare which providers they require so that Terraform can install and use them.



Also check: Types of [Terraform Variables](#)

Terraform AWS Provider

Terraform can provision infrastructure across public cloud providers such as Amazon Web Services (AWS), Azure, Google Cloud, and DigitalOcean, as well as private cloud and virtualization platforms such as OpenStack and VMWare.

AWS is a good choice for learning Terraform because of the following:

- AWS is the most popular cloud infrastructure provider, by far. It has a 45% share in the cloud infrastructure market, which is more than the next [three biggest competitors](#) (Microsoft, Google, and IBM) combined.
- AWS provides a huge range of reliable and scalable cloud hosting services, including Amazon Elastic Compute Cloud (Amazon EC2), which you can use to deploy virtual servers; Auto Scaling Groups (ASGs), which make it easier to manage a cluster of virtual servers; and Elastic Load Balancers (ELBs), which you can use to distribute traffic across the cluster of virtual servers.

- AWS offers a generous Free Tier for the first year that should allow you to run all of these examples for free. If you already used up your free tier credits, the examples in this book should still cost you no more than a few dollars.

Learn more about how to [AWS Free Tier Account](#) to avail the free tier services.

The two most popular options for deploying infrastructure to AWS are [CloudFormation](#), a service native to AWS, and [Terraform](#), an open-source offering from HashiCorp. Most of the AWS resources can be provisioned with Terraform as well and is often faster than CloudFormation when it comes to supporting new AWS features. On top of that, Terraform supports other cloud providers as well as 3rd party services.

Also read: Step by step guide on [Terraform Certification](#)

Azure Terraform Resource Provider

Azure Resource Providers for HashiCorp [Terraform](#) enables Azure customers using Azure Resource Manager (ARM) to provision and manage their resources with Terraform Providers as if they were native Azure Resource Providers.

Below are some of the core infrastructure services supported by Azure Resource Provider in Terraform:

- Virtual machines
- Storage accounts
- Networking interfaces
- [Azure Container Instances](#)
- [Azure Container Service](#)
- [Managed Disks](#)
- [Virtual Machine Scale Sets](#)

The ARM Resource Provider leverages HashiCorp Terraform to provide third-party services to ARM users directly via ARM. Some of these third-party services supported are listed below:

- [Kubernetes](#)
- [Datadog](#)
- [Cloudflare](#)

Terraform is built into [Azure Cloud Shell](#) and cloud shell automatically authenticates your default Azure CLI subscription to deploy resources through the Terraform Azure modules.

To know more about Azure provider for Terraform, click [here](#).

Also Check: Our blog post on [Terraform Commands Cheat Sheet](#). Click here

Oracle Cloud Infrastructure Terraform Provider

Oracle Cloud Infrastructure is an official provider of Hashicorp Terraform supporting infrastructure-as-code for oracle cloud customers.

The provider is compatible with Terraform 0.10.1 and later. Following are some of the main resources supported by the Terraform provider:

- [Block Volumes](#)
- [Compute](#)
- [Container Engine](#)
- [Database](#)
- [File Storage](#)
- [Identity and Access Management \(IAM\)](#)
- [Load Balancing](#)
- [Networking](#)
- [Object Storage](#)

A detailed list of supported resources and more information about how to get started is available on the [HashiCorp website](#).

Oracle also provides **Resource Manager**, a fully managed service to operate Terraform. Resource Manager integrates with Oracle Cloud Infrastructure Identity and Access Management (IAM), so you can define granular permissions for Terraform operations. It also provides state locking, giving users the ability to share state, and lets teams collaborate effectively on their Terraform deployments. Most of all, it makes operating Terraform easier and more reliable.

To know more about Resource Manager, check [here](#).

Oracle had announced two features to help you bring your existing infrastructure to Terraform and Resource Manager:

- **Terraform Resource Discovery** helps you move from a console- or SDK-managed infrastructure to an infrastructure managed by Terraform.
- **State File Import** helps customers who have a Terraform-managed infrastructure move to a Resource Manager-managed infrastructure.

To know more about Terraform Resource Discovery, check [here](#).

Now that we got an overview of what a provider is and services provided by some major providers, let's see how we can use one in our terraform configuration files.

Also Read: Our blog post on [Hashicorp Terraform](#). Click here

Google Cloud (GCP) Terraform Provider

The Google Cloud Terraform Provider is used to configure your [Google Cloud Platform](#) infrastructure. It is collaboratively maintained by the Google Terraform Team at Google and the Terraform team at HashiCorp.

It looks over the lifecycle management of GCP resources, including Compute Engine, Cloud Storage, Cloud SDK, Cloud SQL, GKE, BigQuery, Cloud Functions, and more.

Below is a general provider configuration snippet:

```
provider "google" {
  project      = "my-project-id"
  region       = "us-central1"
}
```

In order to get started with Google Terraform Provider, you need to have a Google Cloud Free Trial Account with billing enabled and terraform installed in your project. Follow the steps from this guide to create your [Google Cloud Free Trial Account](#) and create a project.

Provider Block

Terraform configurations must declare which providers they require so that Terraform can install and use them. Providers are executable plugins that contain the code necessary to interact with the API of the service it was written for.

A provider is defined by a **provider** block, the actual arguments in a provider block vary depending on the provider, but all providers support the meta-arguments of version and alias.

The below image shows the provider block format across different providers.

```
provider azurerm {
  version = "1.41.0"
  tenant_id = var.tenant_id
  subscription = var.subscription_id
}

# Configure the AWS Provider
provider "aws" {
  version = "~> 3.0"
  region  = "us-east-1"
}

# Configure the Oracle Cloud Infrastructure provider with an API Key
provider "oci" {
  tenancy_ocid = "${var.tenancy_ocid}"
  user_ocid = "${var.user_ocid}"
  fingerprint = "${var.fingerprint}"
  private_key_path = "${var.private_key_path}"
  region = "${var.region}"
}
```

Check Out: How to [Install Terraform](#) on Mac, Windows & Ubuntu. Click here

Provider Workflow

Terraform finds and installs providers when initializing a working directory. It can automatically download providers from a Terraform registry, or load them from a local mirror or cache.

Hashicorp distributed providers are available for download automatically during Terraform initialization, while third-party providers must be placed in a local plug-ins directory located at either %APPDATA%\terraform.d\plugins for Windows or ~/.terraform.d/plugins for other operating systems.



The flow of steps performed is explained below:

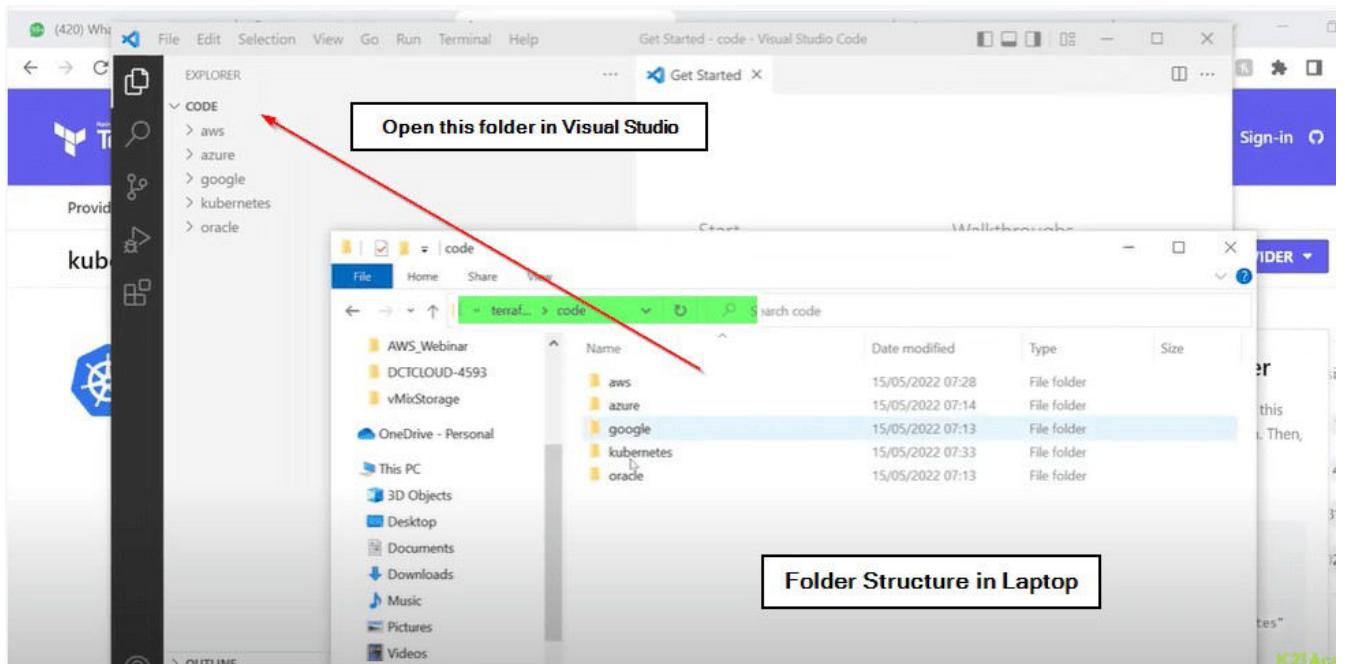
1. Initialize the Terraform configuration, i.e. run `terraform init` command.
2. It looks for provider being used, and download the provider plug-ins, if not found.
3. Retrieves the provider plug-ins
4. Stores them in a `.terraform` subdirectory
5. Check the default version or specified version

How To Use Providers

You can follow some simple steps to use your choice of provider be it AWS, Azure, Google Cloud, or Oracle in any IDE of your choice (here we are using VSCode- Visual Studio Code).

On your laptop, you can create a folder names Terraform and inside this folder create sub-folders like AWS, Azure, Google, etc.

Then open this complete folder in VSCode and also install the HashiCorp Terraform extension.



Now suppose you have to use the AWS provider, then in VSCode inside the AWS folder create a new file as the `provider.tf` and copy the code from the official [AWS provider](#) site in that file. And once done you need to initialize the terraform by running `terraform init` command in the terminal.

Providers / hashicorp / aws / Version 4.17.0 Latest Version

aws

aws

Official by HashiCorp

Public Cloud

Lifecycle management of AWS resources, including EC2, Lambda, EKS, ECS, VPC, S3, RDS, DynamoDB, and more. This provider is maintained internally by the HashiCorp AWS Provider team.

VERSION PUBLISHED SOURCE CODE

4.17.0 5 hours ago hashicorp/terraform-provider-aws

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.17.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

Terraform 0.14+

```
provider "aws" {
  # Configuration options
}
```

EXPLORER

CODE

aws provider.tf

Create a file inside the folder

Get Started provider.tf

aws > provider.tf > provider "aws"

1 terraform {
2 required_providers {
3 aws = {
4 source = "hashicorp/aws"
5 version = "4.14.0"
6 }
7 }
8 }
9
10 provider "aws" {
11 # Configuration options
12 }

PROBLEMS OUTPUT TERMINAL 3 Open Terminal

Directory: C:\terraform\code\aws

Mode	LastWriteTime	Length	Name
-a---	18/05/2022 17:57	171	provider.tf

PS C:\terraform\code\aws> `terraform init` 4

Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "4.14.0"...

K21Academy

Download Terraform

<https://www.terraform.io/downloads.html>

Terraform CLI Documentation

<https://www.terraform.io/docs/cli/index.html>

Course: Using Terraform to Manage Applications and Infrastructure

<https://bit.ly/3Bkr9nT>