Programme Name: BCA and Semester: V
Cloud Computing (BCA50115)
Class: BCA5
Academic Session 2025-2026

Study Material

(Course Name: Cloud Computing and Course Code: BCA50115)
Module II: Cloud Implementations

Table of Contents

Module II: Cloud Implementations

2.1 Cloud deployment models

A deployment model defines the purpose of the cloud and the nature of how the cloud is located. The NIST definition for the four deployment models is as follows:

1. **Public cloud:** The public cloud infrastructure is available for public use alternatively for a large industry group and is owned by an organization selling cloud services.

2. **Private cloud:** The private cloud infrastructure is operated for the exclusive use of an organization. The cloud may be managed by that organization or a third party. Private clouds may be either on- or off-premises.

3. **Hybrid cloud:** A hybrid cloud combines multiple clouds (private, community of public) where those clouds retain their unique identities, but are bound together as a unit. A hybrid cloud may offer standardized or proprietary access to data and applications, as well as application portability.

4. **Community cloud:** A community cloud is one where the cloud has been organized to serve a common function or purpose. It may be for one organization or for several organizations, but they share common concerns such as their mission, policies, security, regulatory compliance needs, and so on. A community cloud may be managed by the constituent organization(s) or by a third party.

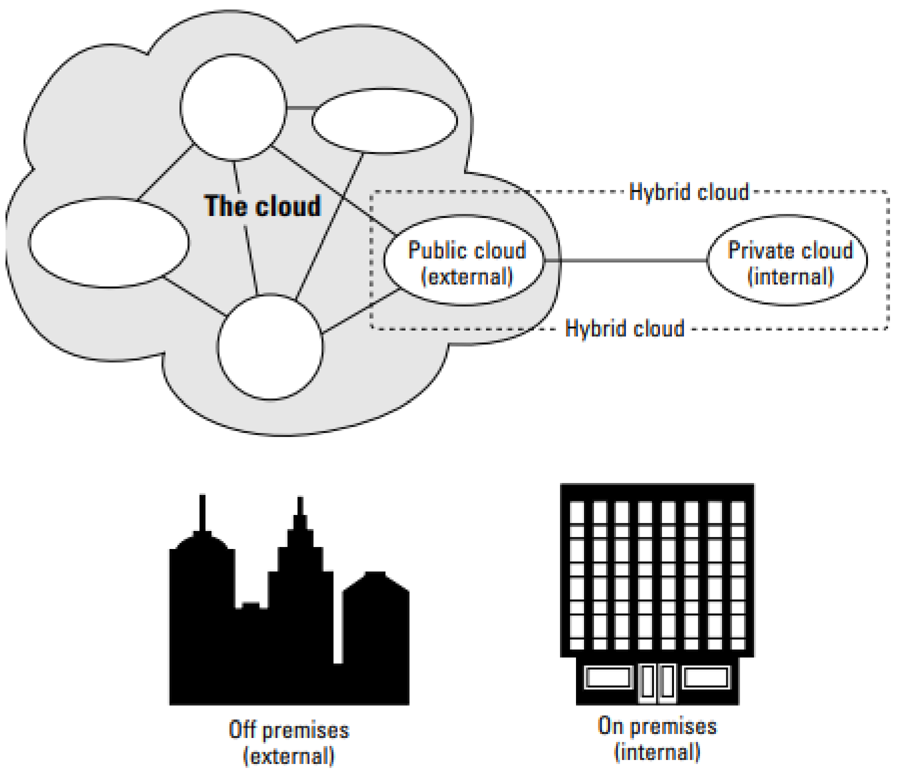
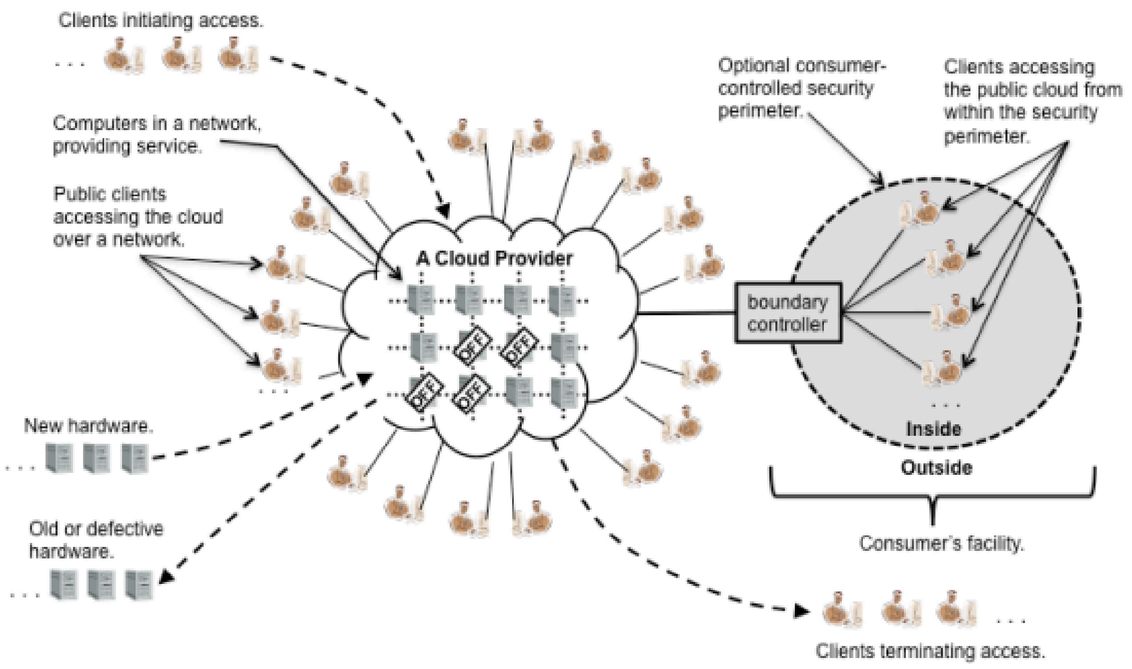


Fig. Deployment locations for different cloud types

2.2 Organizational scenarios of clouds

2.2.1. The Public Cloud Scenario:

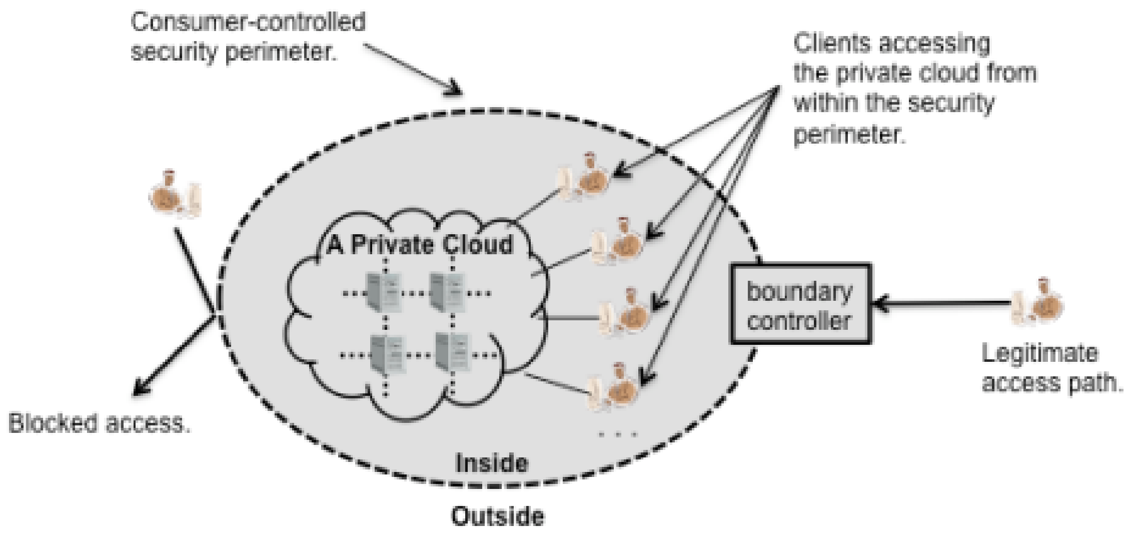
The public cloud scenario involves organizations utilizing cloud resources provided by third-party service providers. These providers offer shared infrastructure and services to multiple clients over the internet. Public cloud platforms, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), provide a wide range of services, including computing power, storage, and networking. Public clouds offer scalability, elasticity, and pay-as-you-go pricing models, allowing organizations to rapidly provision resources and scale their applications based on demand. They are suitable for businesses of all sizes, from startups to large enterprises.


Clients initiating access.
Computers in a network, providing service.
Public clients accessing the cloud over a network.
New hardware.
Old or defective hardware.
A Cloud Provider
OFF
Optional consumer-controlled security perimeter.
Clients accessing the public cloud from within the security perimeter.
boundary controller
Inside
Outside
Consumer's facility.
Clients terminating access.

2.2.2. The Private Cloud Scenario:

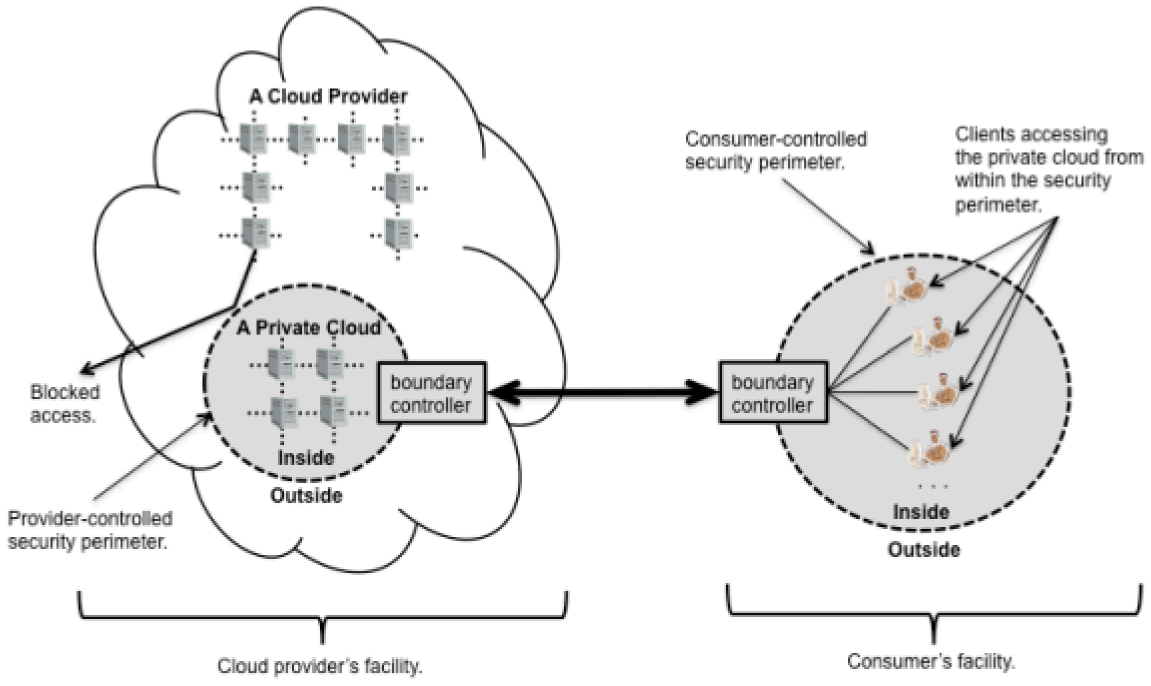
(a) The On-site Private Cloud Scenario:

In the on-site private cloud scenario, organizations deploy and manage their private cloud infrastructure within their own premises. This setup provides complete control and customization over the cloud environment. The organization owns and maintains the necessary hardware, software, and networking infrastructure required for the private cloud. It offers enhanced security, as sensitive data remains within the organization's boundaries. On-site private clouds are suitable for organizations with strict security and compliance requirements, as well as those needing full control over their resources.


Consumer-controlled security perimeter.
Clients accessing the private cloud from within the security perimeter.
A Private Cloud
boundary controller
Legitimate access path.
Blocked access.
Inside
Outside

(b) The Outsourced Private Cloud Scenario:

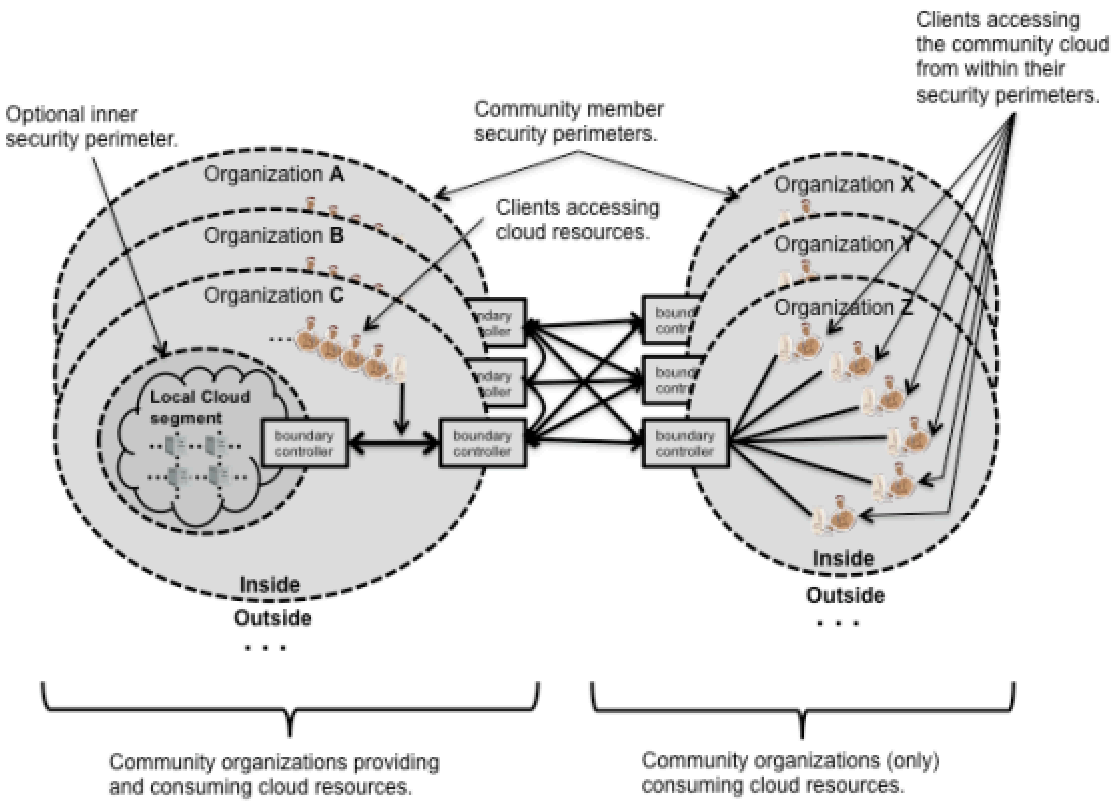
In the outsourced private cloud scenario, organizations partner with a third-party cloud service provider to host and manage their private cloud infrastructure. Instead of investing in on-premises hardware and infrastructure, the organization leases resources from the service provider. The provider takes care of hardware maintenance, software updates, security measures, and other infrastructure-related tasks. This approach offers scalability, flexibility, and reduced operational costs compared to on-site private clouds. Organizations can focus on their core competencies while relying on the expertise of the service provider.


A Cloud Provider
A Private Cloud
boundary controller
Inside
Outside
Blocked access.
Provider-controlled security perimeter.
Consumer-controlled security perimeter.
Clients accessing the private cloud from within the security perimeter.
boundary controller
Inside
Outside
Cloud provider's facility.
Consumer's facility.

2.2.3. The Community Cloud Scenario:

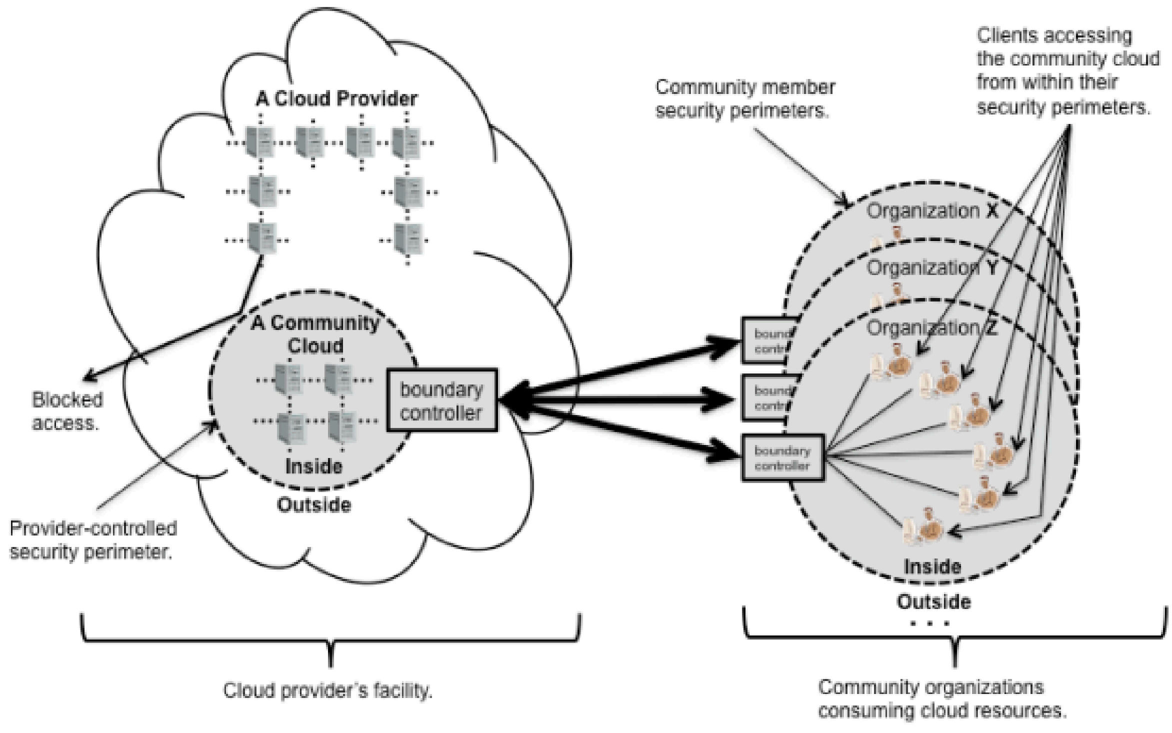
(a) The On-site Community Cloud Scenario:

The on-site community cloud scenario involves multiple organizations with similar interests or requirements sharing a common cloud infrastructure located on their premises. These organizations might belong to the same industry, such as healthcare or education, or have specific regulatory compliance needs. Each organization maintains its data and applications within its private virtual environment, while sharing common resources, such as networking infrastructure and security controls. The on-site community cloud promotes collaboration, resource sharing, and cost optimization among organizations with common goals.


Clients accessing the community cloud from within their security perimeters.
Optional inner security perimeter.
Community member security perimeters.
Organization A
Organization B
Organization C
Clients accessing cloud resources.
Organization X
Organization Y
Organization Z
Local Cloud segment
boundary controller
Inside
Outside
Community organizations providing and consuming cloud resources.
Community organizations (only) consuming cloud resources.

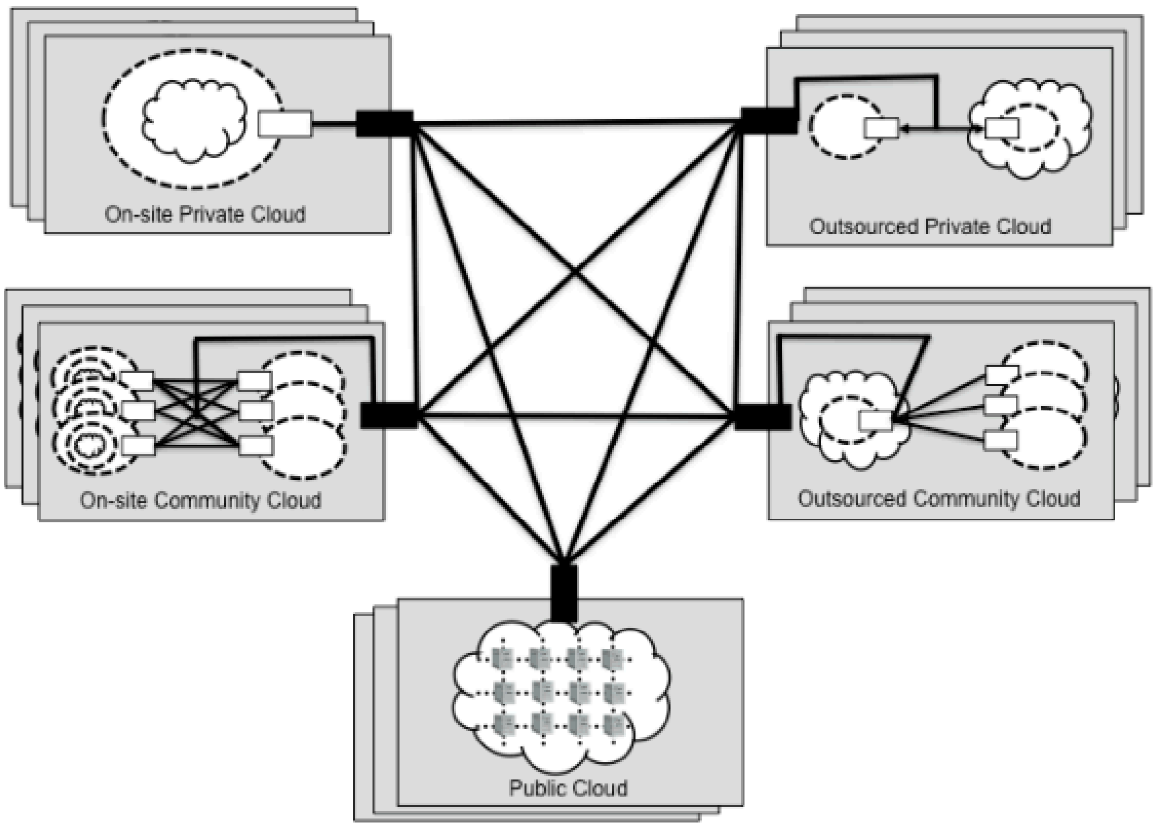
(b) The Outsourced Community Cloud Scenario:

In the outsourced community cloud scenario, multiple organizations with shared interests or requirements outsource their cloud infrastructure to a third-party service provider. This provider specializes in serving communities of organizations with similar needs. The service provider designs and manages the cloud infrastructure, ensuring that each organization's data and applications are isolated and secured. The outsourced community cloud allows organizations to pool their resources, reduce costs, and benefit from shared expertise. It fosters collaboration and knowledge sharing among organizations within the community.


A Cloud Provider
A Community Cloud
boundary controller
Inside
Outside
Blocked access.
Provider-controlled security perimeter.
Community member security perimeters.
Clients accessing the community cloud from within their security perimeters.
Organization X
Organization Y
Organization Z
boundary controller
Inside
Outside
Cloud provider's facility.
Community organizations consuming cloud resources.

2.2.4. The Hybrid Cloud Scenario:

The hybrid cloud scenario combines the use of both public and private cloud environments. Organizations can choose to deploy certain workloads and applications in their private cloud while leveraging the public cloud for others. This approach offers flexibility, allowing organizations to take advantage of the scalability and cost-effectiveness of the public cloud while keeping sensitive data or critical applications in a private cloud for enhanced security and control. Hybrid clouds enable seamless data and application portability between different cloud environments, providing a balance between customization, security, and scalability.



Each of these cloud scenarios offers unique benefits and considerations, and organizations must carefully evaluate their requirements, security needs, and cost considerations to determine the most suitable cloud deployment model for their specific use cases.

2.3 Deploying a web application

Deploying a web application using Remote Desktop on Amazon AWS involves setting up a Windows-based virtual machine (EC2 instance) and configuring it for remote access. Here's an overview of the steps involved:

2.3.1 Launch an EC2 Instance:

i. Sign in to the AWS Management Console.
ii. Open the EC2 service.
iii. Click on "Launch Instance" and select an appropriate Windows Server AMI (Amazon Machine Image).
iv. Choose an instance type, configure the networking, and storage options as per your requirements.
v. Configure the security group to allow inbound connections on the required ports (e.g., port 3389 for Remote Desktop Protocol, RDP).
vi. Launch the instance and generate a key pair if necessary.

2.3.2 Connect to the EC2 Instance:

i. Wait for the EC2 instance to start running.
ii. Obtain the public IP address or DNS name of the instance from the EC2 console.
iii. Use a Remote Desktop Client application (e.g., Microsoft Remote Desktop, Remote Desktop Connection Manager) on your local machine.
iv. Set up a new connection with the instance's public IP/DNS and the appropriate credentials (e.g., username and private key for key pairs or username and password if you created during the instance setup).
v. Connect to the instance.

2.3.3 Set up the Web Application:

i. Once connected to the instance, install any necessary software and dependencies for your web application (e.g., web server, database server, programming language runtime).
ii. Transfer your web application files to the instance using file transfer methods such as FTP, SCP, or by copying files directly to the instance.
iii. Configure Firewall and Security: Adjust the firewall settings on the instance, if required, to allow incoming traffic on the necessary ports for your web application (e.g., port 80 for HTTP, port 443 for HTTPS).
iv. Enable any additional security measures such as SSL certificates for secure communication.

2.3.4 Start the Web Application:

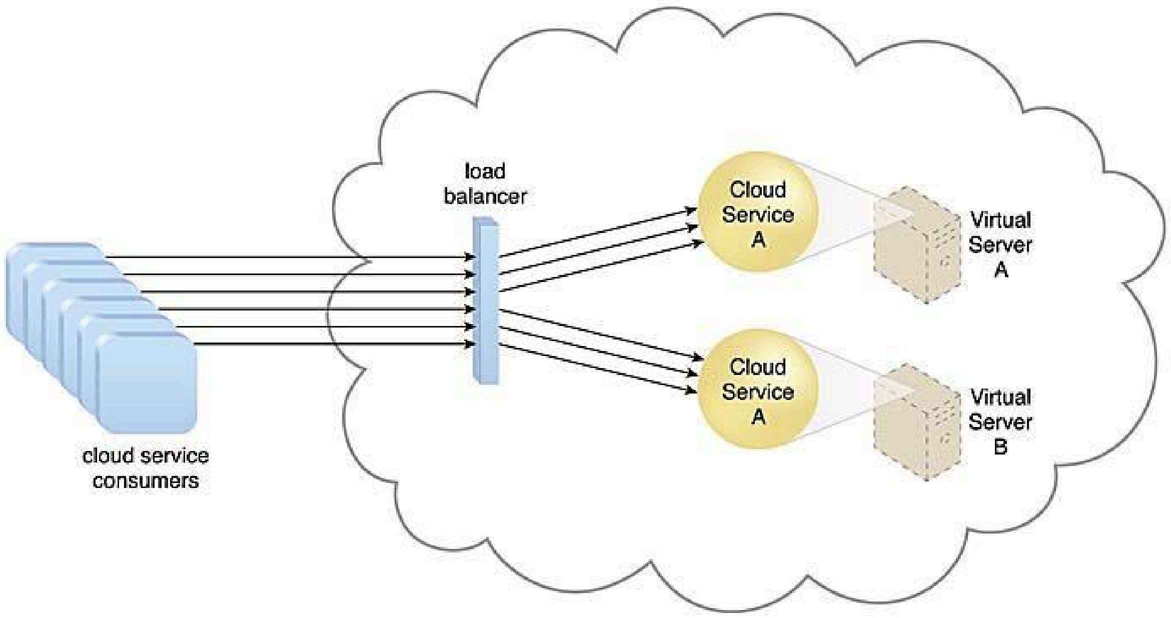
i. Start the web server or any required services for your application.
ii. Ensure that the necessary ports are open and accessible.

2.3.5 Access the Web Application:

i. Use a web browser on your local machine and enter the public IP/DNS of your EC2 instance.
ii. If the web server is running, you should be able to access your web application.

2.4 Workload Distribution

The architecture for workload distribution allows for horizontal scaling of IT resources by adding identical resources and utilizing a load balancer. The load balancer employs runtime logic to evenly distribute the workload among the available resources, effectively reducing both over-utilization and under-utilization of the IT resources. This distribution model can be applied to various IT resources such as distributed virtual servers, cloud storage devices, and cloud services.



Above figure illustrates the implementation of a redundant copy of Cloud Service A on Virtual Server B, with the load balancer intercepting consumer requests and directing them to both Virtual Servers A and B to achieve balanced workload distribution.

Different variations of this architecture exist for specific IT resources, incorporating load balancing aspects. Examples include

1. The service load balancing architecture,
2. The load-balanced virtual server architecture
3. Load-balanced virtual switches architecture

In addition to the base load balancer mechanism, this cloud architecture can include other components:

i. Audit Monitor: Determines if monitoring is necessary based on the type and geographical location of IT resources processing data, to comply with legal and regulatory requirements.
ii. Cloud Usage Monitor: Various monitoring tools are involved in tracking runtime workloads and processing data.
iii. Hypervisor: Distribution of workloads between hypervisors and the virtual servers they host may be required.
iv. Logical Network Perimeter: Isolates cloud consumer network boundaries in relation to workload distribution methods and locations.
v. Resource Cluster: Clustered IT resources in active/active mode are commonly used to support workload balancing between different nodes in the cluster.
vi. Resource Replication: Generates new instances of virtualized IT resources in response to runtime workload distribution demands.

2.5 Resource Pooling

Resource pool is a collection of resources available for allocation to users. All types of resources – compute, network or storage, can be pooled. It creates a layer of abstraction for consumption and presentation of resources in a consistent manner. A large pool of physical resources is maintained in cloud data centers and presented to users as virtual services. Any resource from this pool may be allocated to serve a single user or application, or can be even shared among multiple users or applications. Also, instead of allocating resources permanently to users, they are dynamically provisioned on a need basis. This leads to efficient utilization of resources as load or demand changes over a period of time.

For creating resource pools, providers need to set up strategies for categorizing and management of resources. The consumers have no control or knowledge of the actual locations where the physical resources are located. Although some service providers may provide choice for geographic location at higher abstraction level like- region, country, from where customer can get resources. This is generally possible with large service providers who have multiple data centers across the world.

2.5.1 Resource pooling architecture

Each pool of resources is made by grouping multiple identical resources for example – storage pools, network pools, server pools etc. A resource pooling architecture is then built by combining these pools of resources. An automated system is needed to be established in order to ensure efficient utilization and synchronization of pools.

Computation resources are majorly divided into three categories – Server , Storage and Network. Sufficient quantities of physical resources of all three types are hence maintained in a data center.

(a) Server Pools

Server pools are composed of multiple physical servers along with operating system, networking capabilities and other necessary software installed on it. Virtual machines are then configured over these servers and then combined to create virtual server pools. Customers can choose virtual machine configurations from the available templates (provided by cloud service provider) during provisioning. Also, dedicated processor and memory pools are created from processors and memory devices and maintained separately. These processor and memory components from their respective pools can then be linked to virtual servers when demand for increased capacity arises. They can further be returned to the pool of free resources when load on virtual servers decreases.

(b) Storage Pools

Storage resources are one of the essential components needed for improving performance, data management and protection. It is frequently accessed by users or applications as well as needed to meet growing requirements, maintaining backups, migrating data, etc.

Storage pools are composed of file based, block based or object based storage made up of storage devices like- disk or tapes and available to users in virtualized mode.

1. File based storage – it is needed for applications that require file system or shared file access. It can be used to maintain repositories, development, user home directories, etc.

2. Block based storage – it is a low latency storage needed for applications requiring frequent access like databases. It uses block level access hence needs to be partitioned and formatted before use.

3. Object based storage – it is needed for applications that require scalability, unstructured data and metadata support. It can be used for storing large amounts of data for analytics, archiving or backups.

(c) Network Pools

Resources in pools can be connected to each other, or to resources from other pools, by network facility. They can further be used for load balancing, link aggregation, etc.

Network pools are composed of different networking devices like- gateways, switches, routers, etc. Virtual networks are then created from these physical networking devices and offered to customers. Customers can further build their own networks using these virtual networks.

Generally, dedicated pools of resources of different types are maintained by data centers. They may also be created specific to applications or consumers. With the increasing number of resources and pools, it becomes very complex to manage and organize pools. Hierarchical structure can be used to form parent-child, sibling, or nested pools to facilitate diverse resource pooling requirements.

2.6 Dynamic Scalability

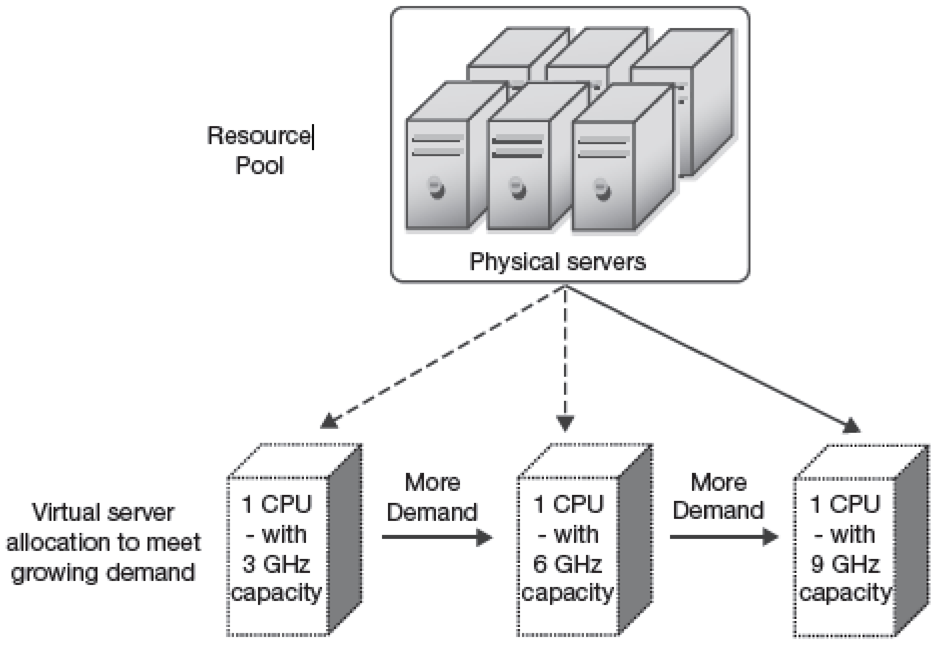
Dynamic scalability refers to the ability of cloud-based resources and services to automatically adjust their capacity based on changing workload demands. It allows organizations to scale their cloud infrastructure up or down in real-time, ensuring optimal resource utilization, performance, and cost-effectiveness.

The dynamic scalability architecture is an architectural model based on a system of predefined scaling conditions that trigger the dynamic allocation of IT resources from resource pools. Dynamic allocation enables variable utilization as dictated by usage demand fluctuations, since unnecessary IT resources are efficiently reclaimed without requiring manual interaction.

2.6.1 Dynamic scalability types

The following types of dynamic scaling are commonly used:

2.6.1 (a) **Vertical Scaling** – One way of increasing resource capacity is to replace the existing component (processors, memory etc.) with a more powerful hardware component. Thus, the capacity can be increased (or decreased) by upgrading (or reducing) a resource quality to support raising (or falling) workload. This type of scaling where a resource component is powered up through replacement is called scaling up or vertical scaling. For example, a system with dual-processor capacity can be scaled vertically by replacing the processor with a quad-processor.

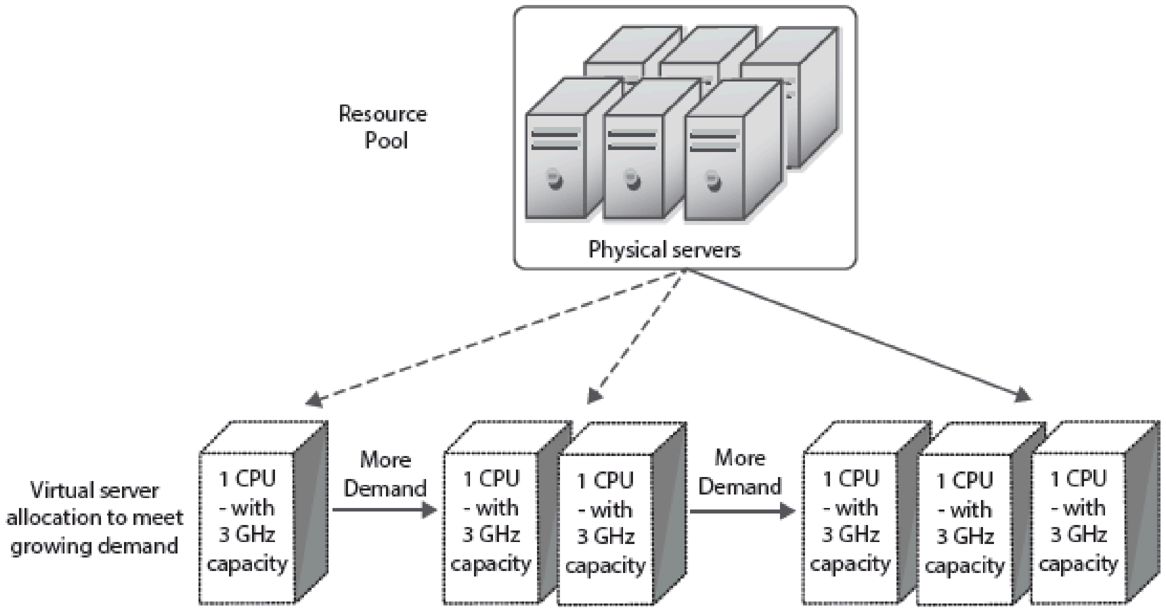

Resource Pool
Physical servers
Virtual server allocation to meet growing demand
1 CPU - with 3 GHz capacity
More Demand
1 CPU - with 6 GHz capacity
More Demand
1 CPU - with 9 GHz capacity

Advantages: It is not a complex task to replace a component of a system since it does not require alteration in the system architecture. So, vertical scaling approach has low management complexity and is less risky.

Disadvantages: High power hardware components or specialized resources are expensive. Capacity or power of any resource component always has an upper limit. Hence, a sufficiently capable resource may not always be available or if available, the specialized hardware components may not be even affordable.

On the other hand, the hardware replacement may cause interruption to the service and thereby introduce a system downtime problem.

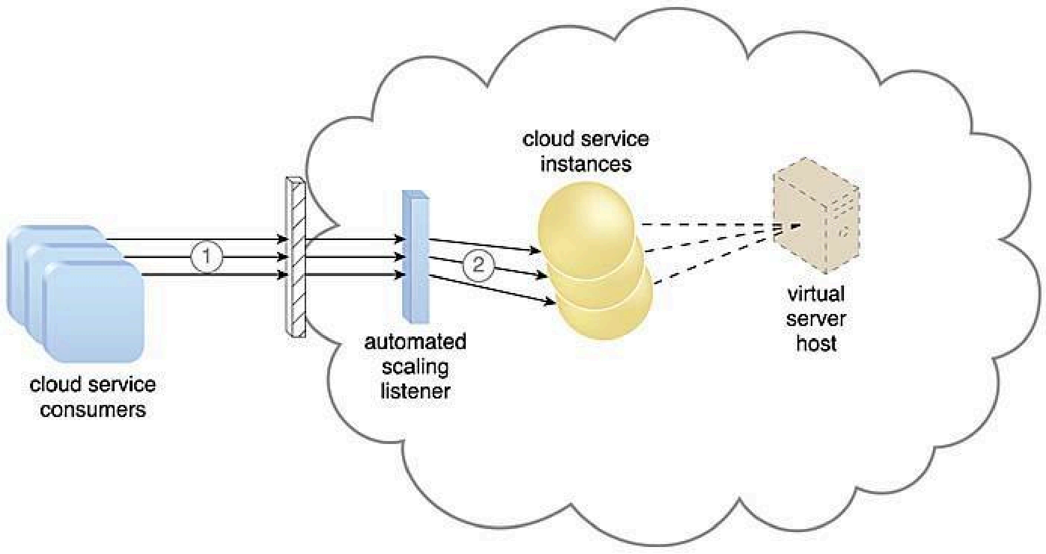
2.6.1 (b) **Horizontal Scaling** – Resource capacity of a system can be increased by introducing additional resources (and without elimination of the existing). These new resources work together with the existing components. This type of scaling where the system is equipped with supplementary resources to expand the resource capacity is known as scaling out or horizontal scaling. For example, a system with a dual-processor capacity can be scaled horizontally by adding another dual-processor.


Resource Pool
Physical servers
Virtual server allocation to meet growing demand
1 CPU - with 3 GHz capacity
More Demand
1 CPU - with 3 GHz capacity
1 CPU - with 3 GHz capacity
More Demand
1 CPU - with 3 GHz capacity
1 CPU - with 3 GHz capacity
1 CPU - with 3 GHz capacity

Advantages: This approach does not involve any system downtime
Disadvantages: This approach needs managing large number of distributed nodes which need to synchronize together, and hence involve complexity.

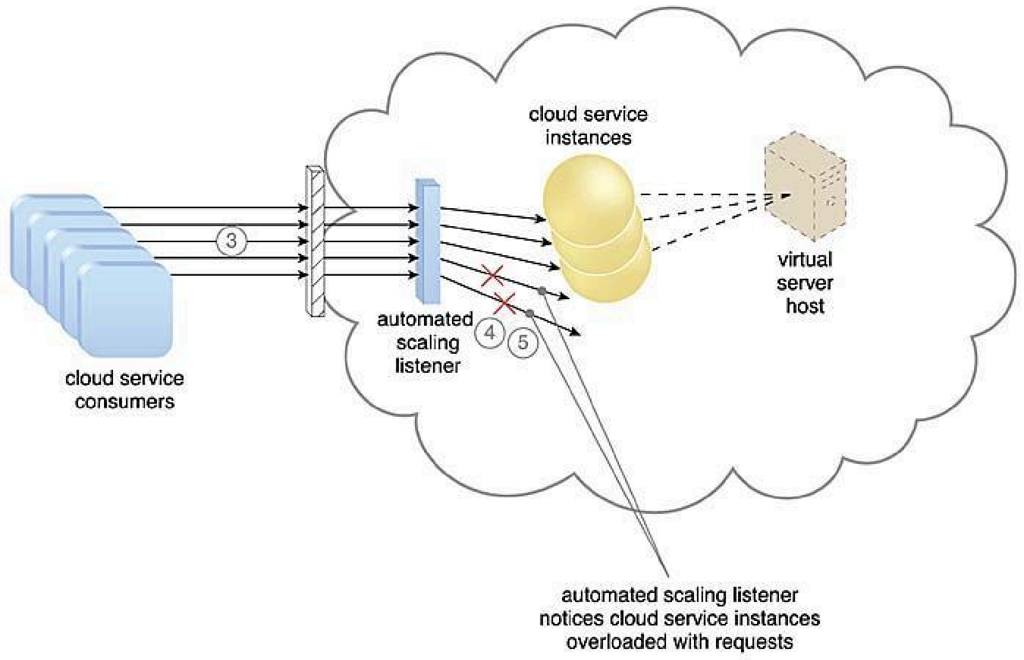
2.6.2 The process of dynamic horizontal scaling:

1. Cloud service consumers are sending requests to a cloud service.
2. The automated scaling listener monitors the cloud service to determine if predefined capacity thresholds are being exceeded.

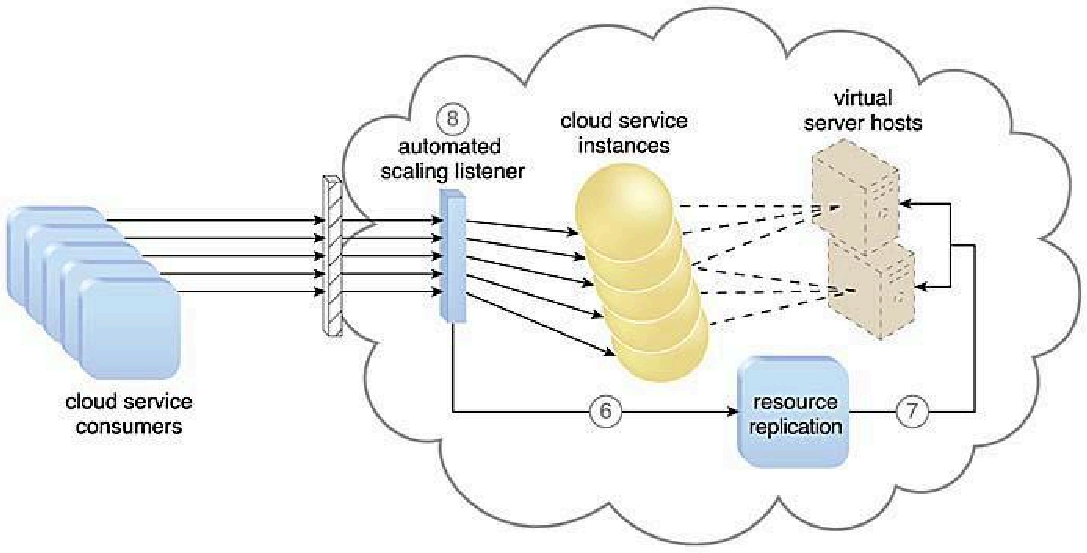


3. The number of requests coming from cloud service consumers increases.

4. The workload exceeds the performance thresholds. The automated scaling listener determines the next course of action based on a predefined scaling policy.
5. If the cloud service implementation is deemed eligible for additional scaling, the automated scaling listener initiates the scaling process.


cloud service instances
virtual server host
3
cloud service consumers
automated scaling listener
4
5
automated scaling listener notices cloud service instances overloaded with requests

6. The automated scaling listener sends a signal to the resource replication mechanism,
7. Resource replication mechanism creates more instances of the cloud service.
8. Now that the increased workload has been accommodated, the automated scaling listener resumes monitoring and detracting and adding IT resources, as required.

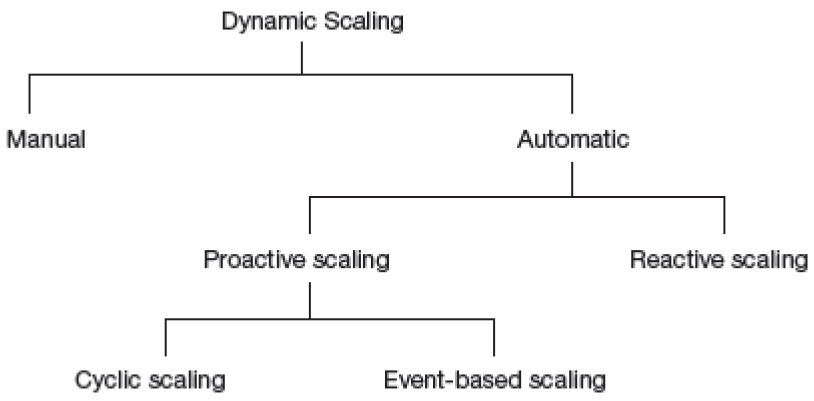


This critical task of dynamic capacity alteration can be done in two ways: ¡) Manually: when a system can be scaled while running by executing appropriate commands through the application interface. ¡¡) Automatically: when this type of scaling of the system can be implemented through programs that can automatically adjust system capacity by observing the actual demand.

The ability of manual capacity adjustment of a system during its operation is a huge task for any computing environment. But passing beyond this, the real power of cloud scaling lies in automatic scaling ability. Here, no human interaction is required. The system can adjust or resize itself on its own.

The dynamic auto-scaling is generally referred as auto-scaling which is also known as cloud scaling. Auto-scaling can be implemented in two different ways: ¡) Scaling based on a predefined schedule known as proactive scaling. ¡¡) Scaling based on current actual demand known as reactive scaling.

The proactive scaling schedules are implemented in two different ways as i) Proactive cyclic scaling: This type of proactive scaling event takes place at fixed regular intervals and by pre-defined times of the day, week, month or year. ii) Proactive event-based scaling: Major variations in traffic load may occur due to some scheduled business events like promotional campaigns or new product launch and else



2.7 Elasticity

Elasticity is defined as the ability of a system to add and remove resources (such as CPU cores, memory, VM and container instances) to adapt to the load variation in real time. Elasticity is a dynamic property for cloud computing.

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.

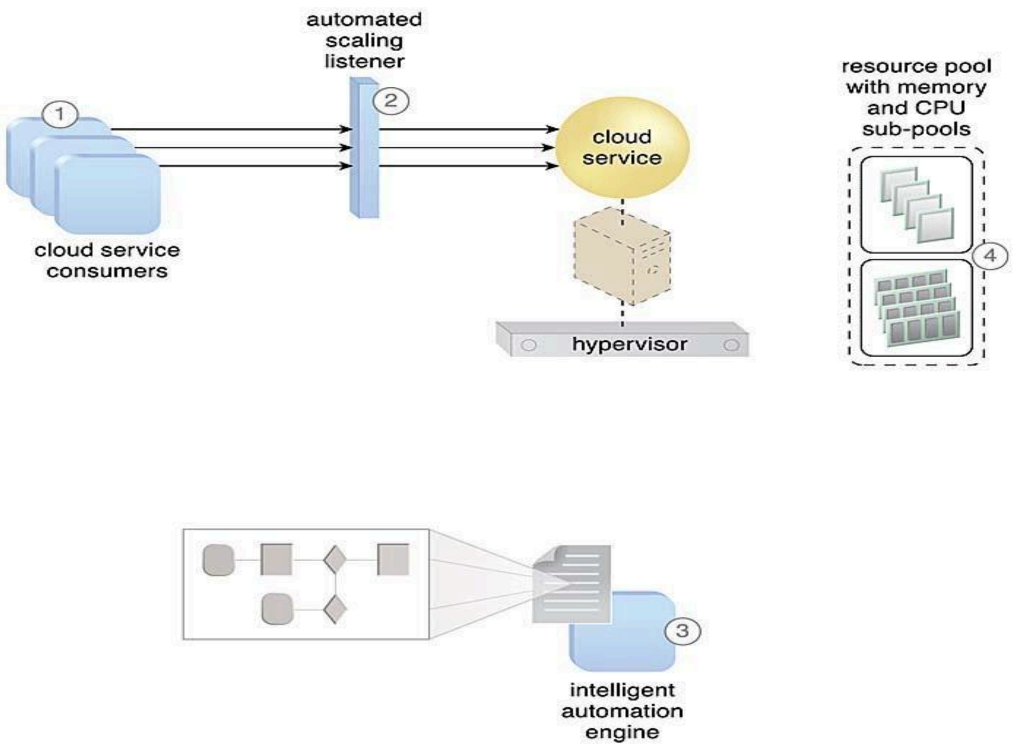
Elasticity = Scalability + Automation + Optimization

Elasticity is built on top of scalability.

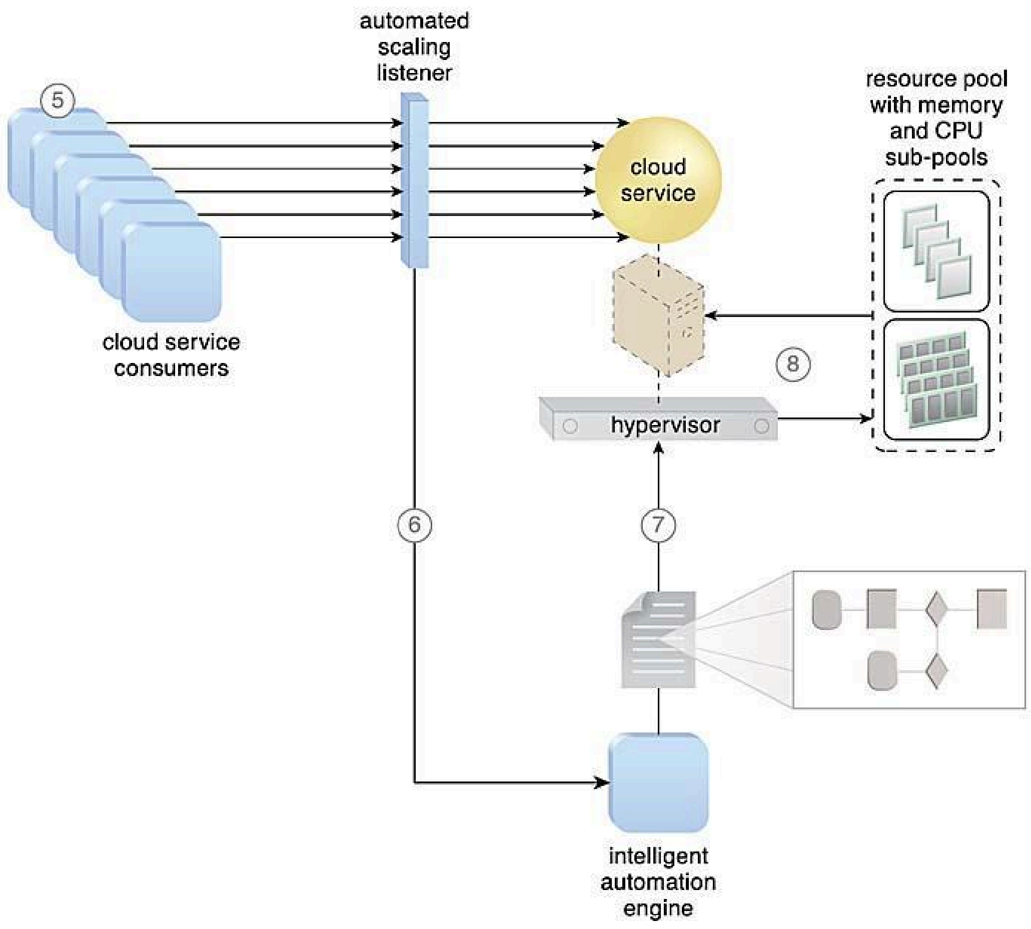
It can be considered as an automation of the concept of scalability and aims to optimize at best and as quickly as possible the resources at a given time.

Mechanism

1. Cloud service consumers are actively sending requests to a cloud service,
2. Requests are monitored by an automated scaling listener
3. An intelligent automation engine script is deployed with workflow logic
4. Automation engine is capable of notifying the resource pool using allocation requests.


automated scaling listener
2
1
cloud service
resource pool with memory and CPU sub-pools
cloud service consumers
4
hypervisor
3
intelligent automation engine

5. Cloud service consumer requests increase.
6. Causing the automated scaling listener to signal the intelligent automation engine to execute the script .
7. The script runs the workflow logic that signals the hypervisor to allocate more IT resources from the resource pools.
8. The hypervisor allocates additional CPU and RAM to the virtual server, enabling the increased workload to be handled.


automated scaling listener
5
resource pool with memory and CPU sub-pools
cloud service
cloud service consumers
8
hypervisor
6
7
intelligent automation engine

Resource pools are used by scaling technology that interacts with the hypervisor and/or VIM to retrieve and return CPU and RAM resources at runtime. The runtime processing of the virtual server is monitored so that additional processing power can be leveraged from the resource pool via dynamic allocation, before capacity thresholds are met. The virtual server and its hosted applications and IT resources are vertically scaled in response.

This type of cloud architecture can be designed so that the intelligent automation engine script sends its scaling request via the VIM instead of to the hypervisor directly. Virtual servers that participate in elastic resource allocation systems may require rebooting in order for the dynamic resource allocation to take effect.

2.8 Service Load Balancing

Load balancing mechanism distributes service requests across cloud applications deployed in data centers spread around the world. Every cloud data centers themselves must have their own load balancers to schedule the incoming service requests towards appropriate resources.

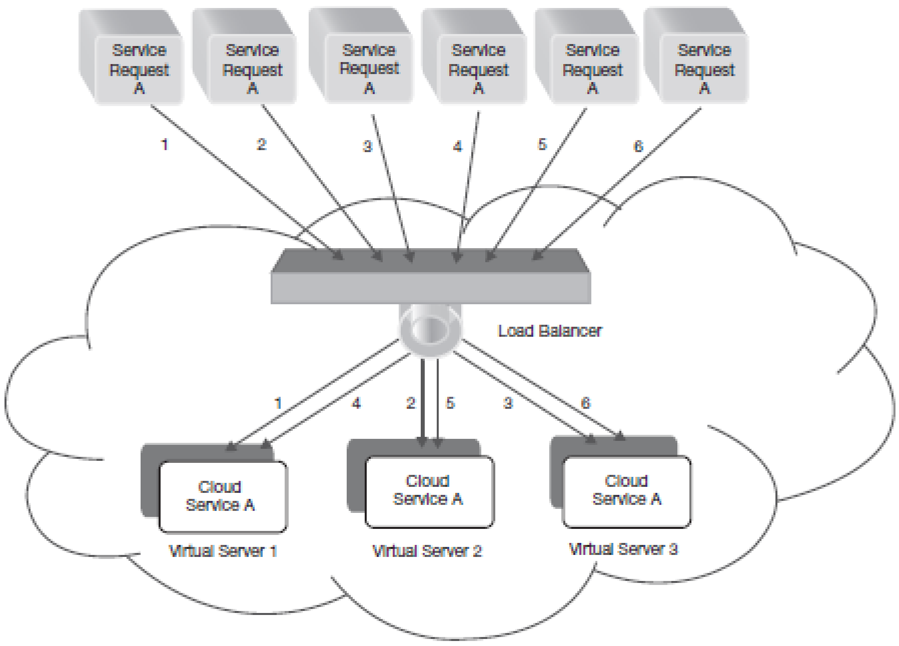
A load-balancing technique can use various tactics for assigning directions to the service requests. In simple form, a load balancer listens to network ports where service requests arrive to identify the kind of application and resources has been requested for. Then, to assign the request to appropriate resource among the available resources, some scheduling algorithm are used.

Depending upon request type and resource requirement to serve the request, different implementations of load balancing mechanisms are required in cloud computing. Among them **service load balancing** is the most critical one. Other requirement types are load balanced virtual switch, load balanced storage mechanism and so on.

In service load balancing, workloads are balanced among multiple instances of each cloud service implementation. The duplicate implementations are organized into a resource pool that responds to fluctuating request volumes. The load balancers can be positioned as either an external or built-in component to allow the host servers to balance the workloads themselves

The load balancer system in a cloud implementation that directly interfaces with clients is called the front-end node. All the incoming requests first arrive in this front end node at the service provider's end. This node then distributes the requests towards appropriate resources for further execution. These resources which are actually virtual machines are called as backend nodes.

In cloud computing implementation, when the load balancers at the front-end node receive multiple requests for a particular service from clients, they distribute those requests among available virtual servers based on some defined scheduling algorithms. This scheduling happens depending on some policy so that all of the virtual servers stay evenly loaded. This ensures maximum as well as efficient utilization of physical and virtual resources. The figure below represents the function of a load balancer in cloud computing environment.



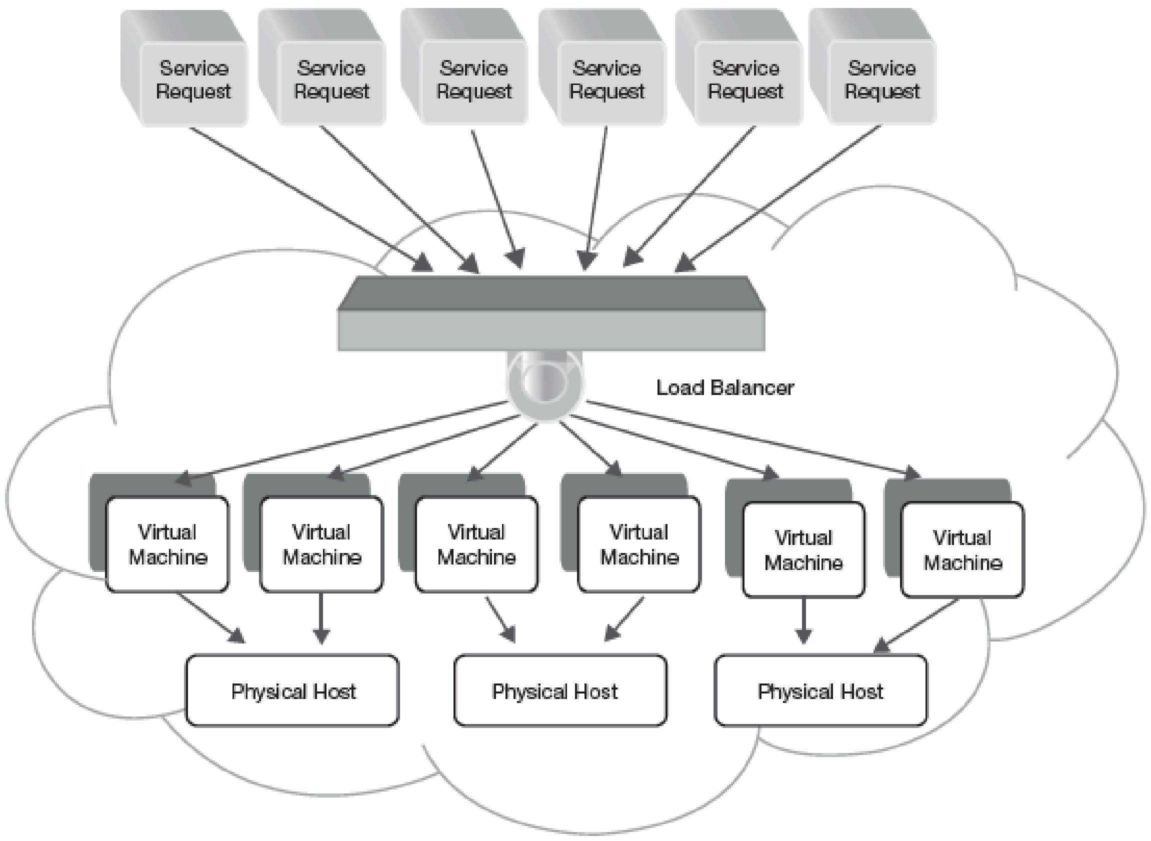
Here it has been assumed that all of these three virtual servers are equally loaded before six similar types of service requests appearing before the system (from same or different consumers). The incoming requests encounter the load balancer first. Load balancers use scheduling algorithm and distribute the requests among these three available virtual servers.

Here, the load balancers act as the front-end-nodes for all of the incoming requests and the virtual servers act as the back-end-nodes.

2.8.1 Two Levels of Balancing: VM Provisioning and Resource Provisioning

Cloud provisioning is the technique of allocating cloud provider's resources to a customer. The automated provisioning mechanism is realized through load balancing activity. Once the front-end-node redirects the incoming service requests, all of other activities happen at the back-end-nodes.

The assignments and executions of application requests at the back-end-nodes happen in two phases as shown in figure below. In the first step, a service request is analyzed through its characteristics by the load balancer and assigned to an appropriate virtual machine instance which is called **VM provisioning**. In the second step, these requests are mapped and scheduled onto actual physical resources, and this is known as **resource provisioning**



2.8.2 Goals of Load Balancing

The primary goal of load balancing is to distribute service loads among resources. Apart from this, there are other objectives also. Following section briefly narrates these objectives:

To improve system performance: In a load balanced system, no resource should get overloaded when others are under-utilized. All resource components remain almost evenly loaded or uniformly free. This improves the performance and stability of the overall system.

To maximize fault tolerance: In a load balanced system, while multiple nodes work together, that enhances the tolerance of the system against the faults. If some node stops functioning due to any kind of hardware or software failure the traffic is automatically redirected among other working nodes by keeping the system up. Thus, the users of such system remain unaffected and unaware of the failure.

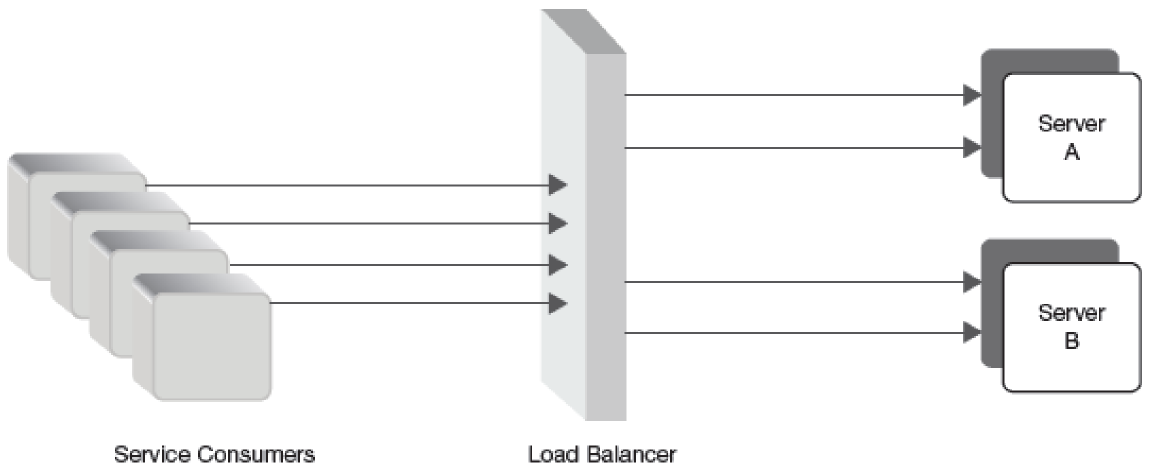
To accommodate scaling: An efficient computing system needs to scale as application demand grows or declines. One objective of load balancing is to support scaling by properly redirecting load to the newly introduced nodes in case of growing or by releasing load from nodes in case of shrinking.

To ensure availability of applications all the time: Another objective of load balancing is to keep the applications available all the time. As higher level of fault tolerance can be achieved.

2.8.3 TYPES OF LOAD BALANCING

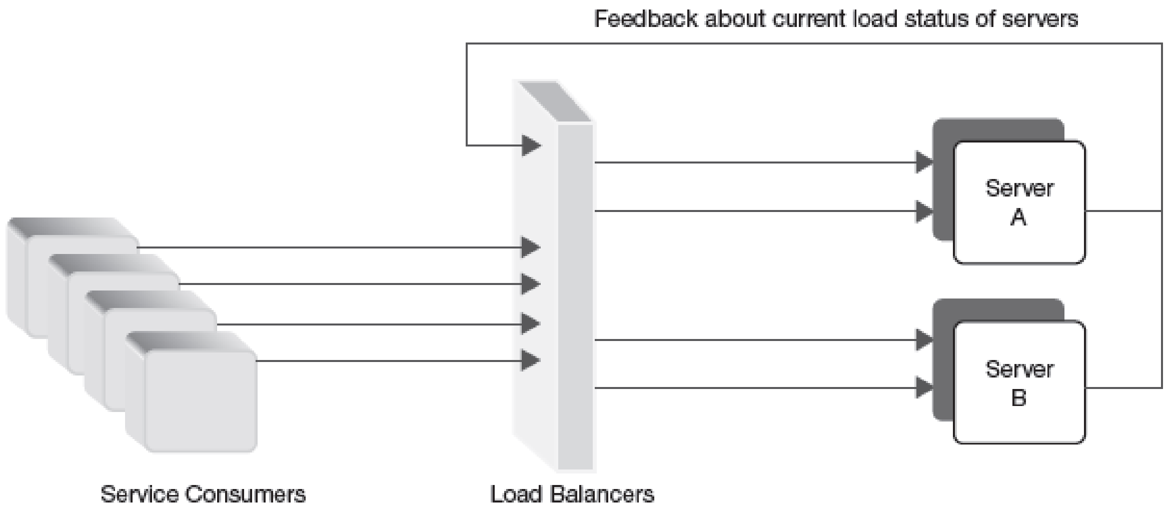
For efficient resource utilization, as well as for effective task execution, a load balancer should have knowledge about the engagements of resources while assigning tasks to resources. Depending upon the use of knowledge base, there are two categories of load balancing technique: static and dynamic.

Static load balancing does not use any knowledge base and distributes service requests only based on the characteristics of the requests it is assumed that all of the incoming service requests are similar in nature. It is further assumed that server A and server B are both able to handle those requests. Static load balancing algorithms divide the incoming traffic evenly among two available servers. It only ensures that all of the servers are receiving almost equal number of service requests based on the specific features of requests



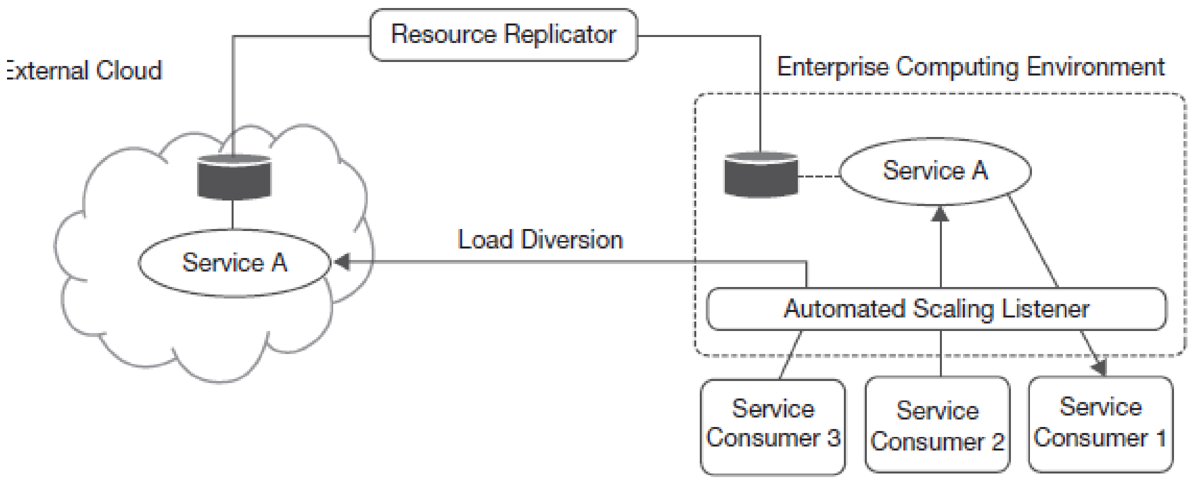
The **dynamic approach of load balancing** technique does not distribute loads among resources depending on any fixed set of rules. Scheduling of tasks usually happen based on the existing state of the resources or nodes. The advantage of using dynamic load balancing is that if any node gets fully loaded or fails, it will not halt the whole system.

This approach is implemented by applying feedback mechanism where the resource components are monitored continually. The nodes periodically send load and status information to the load balancer and if any node becomes non-responsive during operation, the load balancer stops sending traffic to it.



2.9 Cloud bursting

The foundation of the cloud bursting architectural model is based on two functionalities like automated scaling listener and resource replicator. The automated scaling listener decides when to redirect load to an external cloud system. Resource replication maintains the system status during load switch between on-premises system and cloud environment. Figure below represents the architectural model of cloud bursting.



As shown in Figure above, the automated scaling listener monitors the load of on-premises service 'A'. When on-premises service load exceeds the threshold value, the scaling listener redirects traffics towards the replicated implementation of service 'A' in the external cloud.

In the meantime, the resource replicator copies the current state and data of on-premises service ‘A’ to external cloud in order to keep the systems synchronized. Service ‘A’ remains redundantly pre-deployed in the external cloud in an inactive mode until cloud bursting occurs.

In Figure above , when the on-premises service ‘A’ becomes fully-loaded with traffics from service consumers 1 and 2, the automated scaling listener redirects the traffics from service consumer 3 towards the external cloud implementation of Service ‘A’. When load decreases, the external cloud service is released and the system retreats to the internal on-premises system.

2.10 **Service Technology: SOAP, Service Middleware**

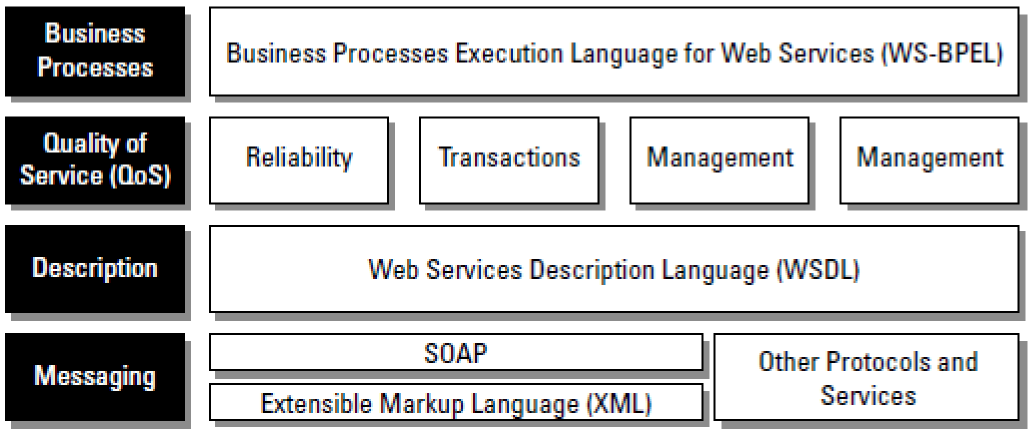
2.10.1 Introduction:

Service-Oriented Architecture (SOA) is a standardized method for requesting services from distributed components and managing the results. SOA removes barriers for clients to obtain desired services by allowing them to be written in different languages and use various messaging and networking protocols for communication. It provides access to reusable Web services over a TCP/IP network, which is important for cloud computing.

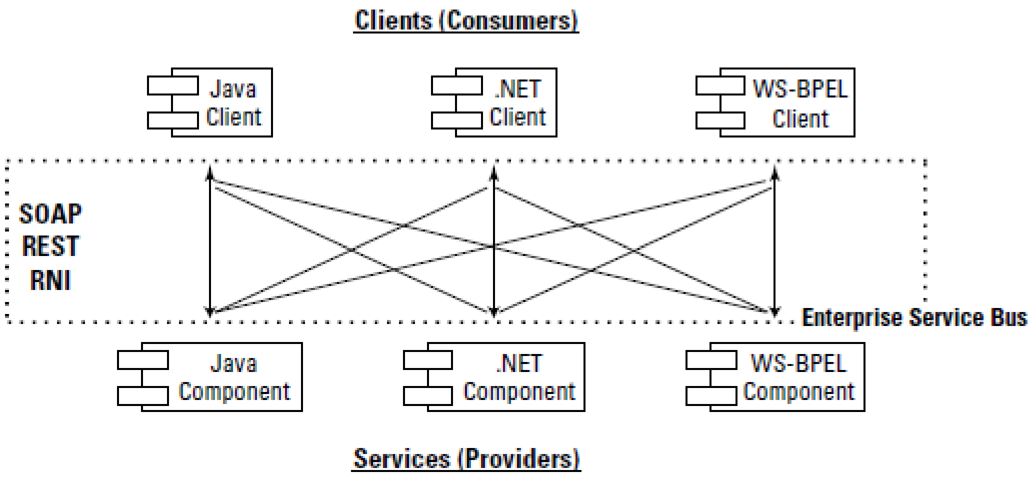
While monolithic cloud applications with specific functions don't require SOA, as cloud computing expands to offer diverse services, SOA becomes valuable for accessing ready-made, modular, and optimized components, minimizing costs for developers and infrastructure. Over the past decade, SOA has been a collaborative effort among vendors to create a common solution for architecting complex business software processes efficiently.

SOA is an architecture that creates a virtual message-passing system with loose coupling between clients and services. **Middleware** software plays a crucial role in SOA implementations, serving as a transaction manager and translator. Universal Description Discovery and Integration (UDDI) is commonly used to discover available web services.

SOA describes a message-passing taxonomy for a component-based architecture where clients pass messages containing metadata to a component, which acts upon the message and returns a response. **XML with SOAP is a commonly used message-passing format**, but other formats like WSDL, WSS, and WS-BPEL are possible. SOA allows creating integrated processes as a set of linked services, and components expose themselves as "endpoints" to clients. A protocol stack for SOA showing the relationship of each protocol to its function is given below:

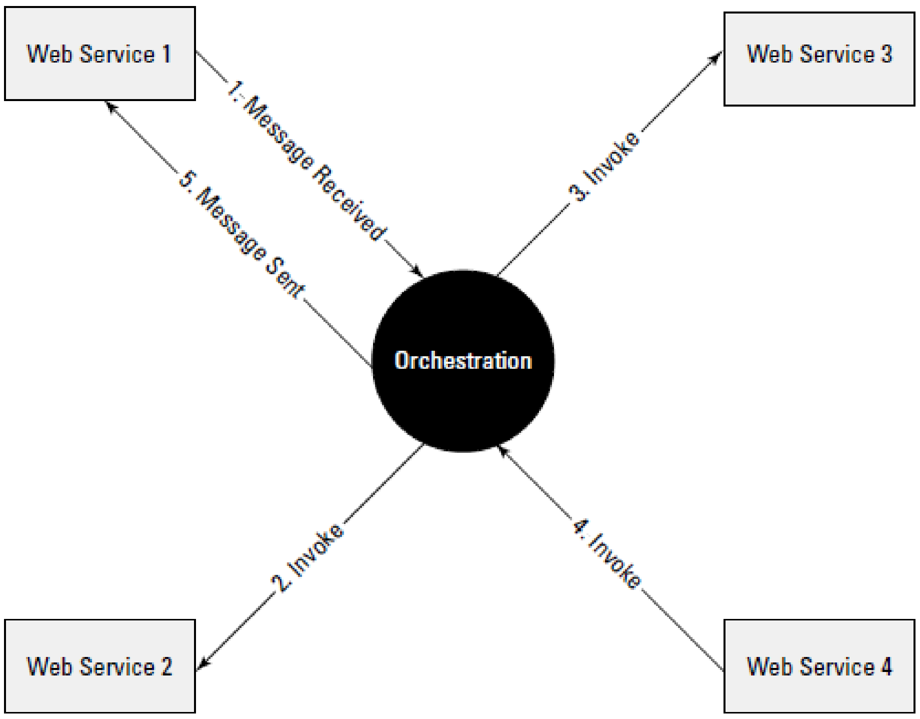

Business Processes
Business Processes Execution Language for Web Services (WS-BPEL)
Quality of Service (QoS)
Reliability
Transactions
Management
Management
Description
Web Services Description Language (WSDL)
Messaging
SOAP
Extensible Markup Language (XML)
Other Protocols and Services

SOA allows for different component and client construction, as well as access to each using different protocols



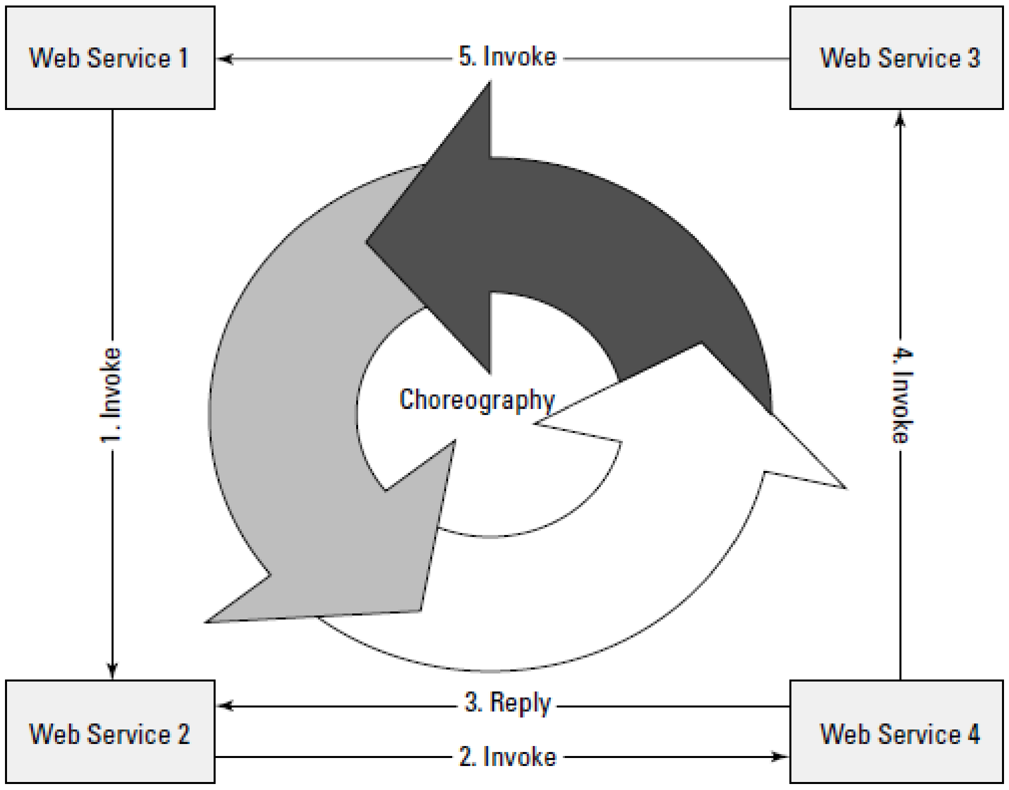
Service Orchestration and Service Choreography are both important concepts in Service-Oriented Architecture (SOA) as they play distinct roles in managing and coordinating web services within a distributed system. Each approach addresses different aspects of service integration and brings unique benefits to the architecture.

In orchestration, a middleware service centrally coordinates all the different Web service operations, and all services send messages and receive messages from the orchestrator. The logic of the compound business process is found at the orchestrator alone.



One way of performing orchestration is through the use of an Enterprise Service Bus or ESB. An ESB provides a middleware software layer for event management with a messaging infrastructure.

A compound business process that uses choreography has no central coordination function. In choreography, each Web service that is part of a business process is aware of when to process a message and with what client or component it needs to interact with. Choreography is a collaborative effort where the logic of the business process is pushed out to the members who are responsible for determining which operations to execute and when to execute them, the structure of the messages to be passed and their timing, and other factors. With choreography, business process execution is a cooperative affair.



2.10.2 Service Technology

Service technology refers to the set of protocols, tools, and middleware used to facilitate communication, integration, and interaction between different services and components within the cloud environment. Service technology is necessary to implement the concepts and principles of SOA effectively.

The role of service technology in SOA includes:

Communication Protocols: Service technology, such as SOAP, REST, or other communication protocols, enables services to exchange data and messages, facilitating seamless communication between different components.

Middleware: Middleware software provided by Enterprise Service Bus plays a critical role in service technology by providing a set of services and functionalities for managing communication, routing messages, handling security, and ensuring the proper execution of business processes.

Message Formats: Service technology defines the format of messages exchanged between services, such as XML or JSON, allowing for standardized data representation and understanding among different services.

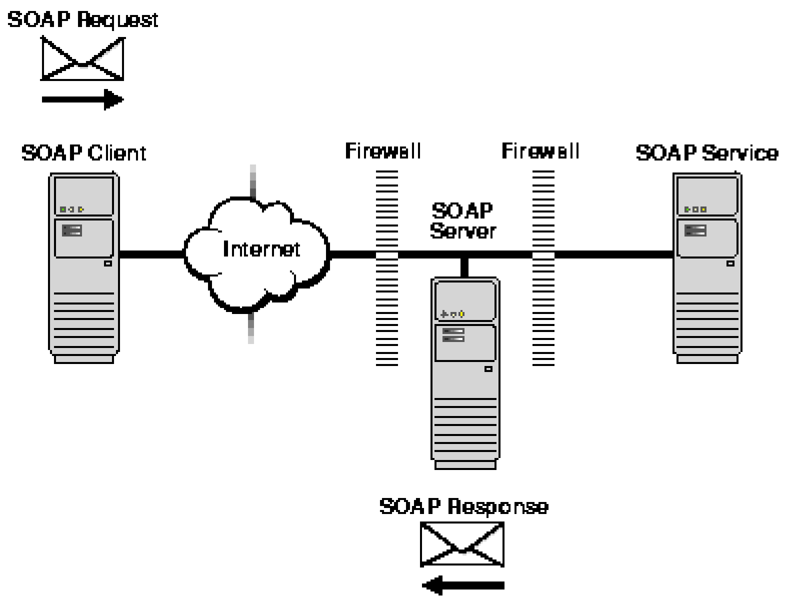
Service Discovery: Service technology enables the discovery of available services and their capabilities, making it possible for services to dynamically locate and interact with each other.

Service-Oriented Architecture (SOA) lays the conceptual groundwork for organizing services in a flexible and reusable manner, while service technology provides the practical tools and middleware to facilitate communication, integration, and interaction between these services within the cloud environment.

2.10.3 Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) is an application protocol developed in 1998 for Web applications; its message format is based on the Extensible Markup Language (XML). The SOAP is a platform and the language independent communication protocol. Services in SOA paradigm communicate through this protocol. SOAP uses TCP and, more recently, UDP transport protocols. It can also be stacked above other application layer protocols such as HTTP, SMTP, or JMS. The platform independence enables the SOAP to establish communication between applications running on different operating systems and the property of language independence enables it to establish communication between applications developed in different programming languages. The processing model of SOAP is based on a network consisting of senders, receivers, intermediaries, message originators, ultimate receivers, and message paths. SOAP is an underlying layer of Web Services.

The components in the SOAP architecture is given below.



Note: The figure above shows the path of a SOAP request and a SOAP response between the SOAP Client and the SOAP Server. The SOAP request is sent from the system running the SOAP Client, going out over the Internet as a SOAP request. The SOAP server receives the request, possibly through a firewall, and passes the service request to the SOAP Service. The SOAP Server then sends the SOAP response over the Internet and back to the SOAP client.

In general, a SOAP service remote procedure call (RPC) request/response sequence includes the following steps:

1. A SOAP client formulates a request for a service(creating a conforming XML document)
2. A SOAP client sends the XML document to a SOAP server. This SOAP request is posted using HTTP or HTTPS to a SOAP Request Handler running as a servlet on a Web server.
3. The Web server receives the SOAP message, an XML document, using the SOAP Request Handler Servlet. The server then dispatches the message as a service invocation to an appropriate server-side application providing the requested service.

4. A response from the service is returned to the SOAP Request Handler Servlet and then to the caller using the standard SOAP XML payload format.

2.10.4 Service middleware

Service middleware refers to a category of software that acts as an intermediary layer between different components and services within a Service-Oriented Architecture (SOA). It facilitates communication, integration, and interaction among services, making it a crucial component in building scalable, flexible, and loosely coupled distributed systems.

The role of service middleware in SOA includes:

Communication Management: Service middleware handles the communication between various services, abstracting the complexities of network communication. It provides a unified interface for services to interact with each other, regardless of their underlying platforms or technologies.

Message Transformation: Middleware can handle data format conversions, allowing services with different data formats to communicate seamlessly. It ensures that data exchanged between services is properly translated and understood.

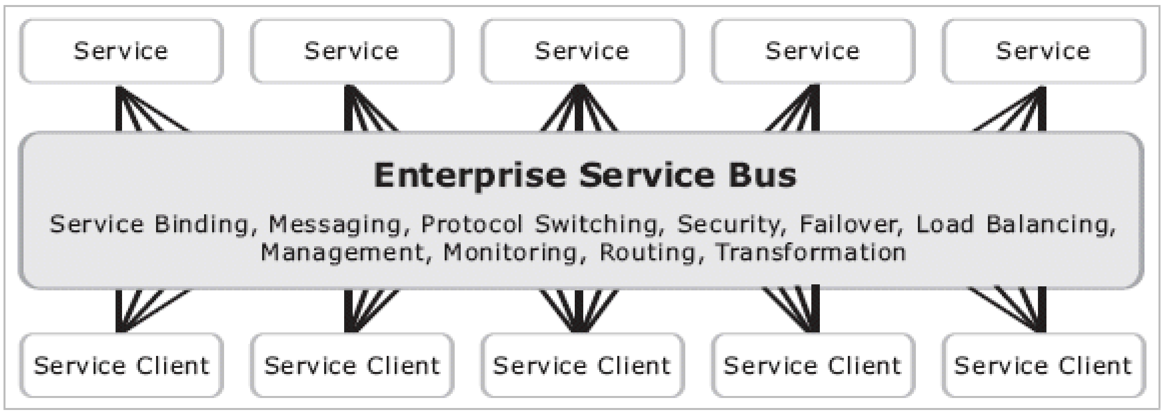
Service Virtualization: Middleware abstracts the underlying complexity of service locations and interfaces, allowing services to be exposed and accessed in a standardized manner. This service virtualization enhances the flexibility and reusability of services within the SOA.

Security and Authentication: Service middleware enforces security policies and provides authentication and authorization mechanisms to secure communication between services. It acts as a gatekeeper, ensuring that only authorized services can access specific resources.

Routing and Orchestration: Middleware can manage the routing of messages between services and handle service orchestration to create complex business processes. It ensures that messages are delivered to the appropriate services based on predefined rules.

Enterprise Service Bus (ESB):

An Enterprise Service Bus (ESB) is a specific type of service middleware that plays a central role in SOA. It is an integration framework that provides a comprehensive and standardized approach to connect and manage services within an enterprise or distributed system. The ESB acts as a message broker, facilitating communication and data exchange between services, applications, and systems.


Service
Service
Service
Service
Service
Enterprise Service Bus
Service Binding, Messaging, Protocol Switching, Security, Failover, Load Balancing, Management, Monitoring, Routing, Transformation
Service Client
Service Client
Service Client
Service Client
Service Client

In its most basic form, an ESB offers the following key features:

- Web services: support for SOAP, WSDL and UDDI, as well as emerging standards such as WS-Reliable Messaging and WS-Security
- Messaging: asynchronous store-and-forward delivery with multiple qualities of service
- Data transformation: XML to XML
- Content-based routing: publish and subscribe routing across multiple types of sources and destinations
- Platform-neutral: connect to any technology in the enterprise, such as Java, .Net, mainframes, and databases

In addition to ESB, there are other service middleware technologies and frameworks that support SOA principles, such as:

1. Apache Camel: Apache Camel is an open-source integration framework that provides routing and mediation capabilities, making it suitable for building ESB-like solutions.

2. MuleSoft Anypoint Platform: Anypoint Platform is a comprehensive integration platform that includes ESB capabilities. It offers API management, data integration, and messaging features to facilitate communication between services.

3. Apache ServiceMix: Apache ServiceMix is an open-source ESB and integration platform based on Java technologies. It provides features like service virtualization, messaging, and routing.

4. WSO2 ESB: WSO2 ESB is an open-source enterprise service bus that supports service mediation, routing, and transformation to enable seamless communication and integration between services.

Questions

MCQ

1. What is the main characteristic of the Public Cloud scenario?
    a) Complete control and customization over the cloud environment.
    b) Shared infrastructure and services provided by third-party providers.
    c) Collaboration and cost optimization among organizations.
    d) Combining public and private cloud environments.

2. What does dynamic scalability refer to?
    a) Enhanced security with sensitive data within the organization's boundaries.
    b) Sharing common resources with other organizations
    c) The automatic adjustment of resource capacity based on changing workload demands.
    d) Complete control and customization over the cloud environment.

3. Which type of scaling allows the system to automatically adjust its capacity based on actual demand?
    a) Proactive scaling.
    b) Reactive scaling.
    c) Vertical scaling.
    d) Horizontal scaling.

4. Elasticity in cloud computing refers to:
    a) The ability of a system to adapt to load variation in real time
    b) The ability to deploy multiple VMs in parallel
    c) The capability of a system to scale vertically
    d) The process of deploying resources manually in response to demand

5. In a cloud architecture with elastic resource allocation, what triggers the hypervisor to allocate more IT resources?
    a) Cloud service consumers actively sending requests
    b) The intelligent automation engine executing the script
    c) The cloud service consumer requests increasing
    d) The automated scaling listener monitoring the requests

6. Load balancing in cloud computing is essential for:
    a) Distributing resources evenly among servers
    b) Reducing the number of virtual machines
    c) Managing data storage effectively
    d) Distributing service requests across cloud applications

7. What is the role of Service Technology in Service-Oriented Architecture (SOA)?
    a) Providing a set of services and functionalities for managing communication, routing messages, and handling security.
    b) Coordinating all the different Web service operations centrally.
    c) Defining the format of messages exchanged between services.
    d) Creating a virtual message-passing system with loose coupling between clients and services.

8. Which of the following is an example of a communication protocol used in Service Technology?
    a) SOAP
    b) UDDI
    c) JSON
    d) WSDL

9. What is the main difference between Service Orchestration and Service Choreography in Service-Oriented Architecture (SOA)?
    a) Orchestration involves a centralized coordination function, while choreography is a cooperative affair with no central coordination.
    b) Orchestration uses middleware like ESB, while choreography relies on communication protocols like REST.
    c) Orchestration is used in monolithic cloud applications, while choreography is essential for diverse cloud services.
    d) Orchestration is used for client construction, while choreography is used for component construction.

10. Which of the following is a specific type of service middleware that plays a central role in SOA?
    a) Apache Camel
    b) MuleSoft Anypoint Platform
    c) SOAP
    d) Enterprise Service Bus (ESB)

Short Questions (3 marks each):

1. What are the two main deployment scenarios of the Private Cloud model?
2. Describe the benefits of the Community Cloud scenario.
3. Write about types of dynamic scaling commonly used?
4. Define elasticity in the context of cloud computing.
5. Differentiate between service orchestration and service choreography in the context of SOA.
6. Write about types of load balancing in cloud.
7. Explain cloud bursting with diagram.
8. Write about role of service middleware in SOA.
9. What is dynamic scalability. Write about horizontal scaling.
10. Write about goals of load balancing.

Long Questions (5 marks each):

1. What is workload distribution, discuss mechanism of workload distribution with diagram.
2. Compare and contrast the Public Cloud and Private Cloud scenarios, highlighting their respective advantages and use cases.
3. Explain the architecture of resource pooling in cloud computing and how it ensures efficient utilization of resources.
4. Explain mechanism of dynamic horizontal scaling with appropriate diagram.
5. Describe the architectural model of cloud bursting and the role of the automated scaling listener and resource replicator.
6. Discuss the significance of service middleware in SOA and elaborate on the features of the Enterprise Service Bus (ESB).

Questions for advanced learner

1. Describe the process of deploying an application over a cloud infrastructure, discussing the essential steps involved and the benefits of this approach.
2. Explain the concepts of Workload Distribution and Resource Pooling in cloud computing. How do these concepts contribute to optimizing resource utilization and enhancing performance?
3. What is dynamic scalability in the context of cloud services? Discuss how it helps organizations to efficiently adapt to changing workload demands.
4. Elaborate on the concept of elasticity in cloud computing. How does it address the challenge of resource provisioning and support varying workloads effectively?
5. Define Service Load Balancing and its significance in cloud environments. How does it ensure fair distribution of incoming service requests across multiple servers?
6. Describe the concept of Cloud Bursting and its role in managing peak workloads. How does it enable seamless expansion from private to public clouds?
7. Compare Service-Oriented Architecture (SOA) and SOAP (Simple Object Access Protocol) in the context of cloud service technology. How do they relate to each other in cloud-based applications?
8. Explain the importance of Service Middleware in cloud computing systems. How does it enable communication and interaction between distributed services?
9. Discuss the challenges and considerations organizations should take into account when deciding to adopt a specific cloud deployment model for their applications.

Answers to the MCQs:

1	b) Shared infrastructure and services provided by third-party providers.
2	c) The automatic adjustment of resource capacity based on changing workload demands.
3	b) Reactive scaling.
4	a) The ability of a system to adapt to load variation in real time.
5	c) The cloud service consumer requests increasing.
6	d) Distributing service requests across cloud applications.
7	A) Providing a set of services and functionalities for managing communication, routing messages, and handling security.
8	A) SOAP
9	A) Orchestration involves a centralized coordination function, while choreography is a cooperative affair with no central coordination.
10	D) Enterprise Service Bus (ESB)