Unit – 2 cloud computing architecture [bim]

2.1 Cloud Computing Reference Model

The in-demand availability of computer system resources, particularly data storage and processing power, without the user's direct involvement is known as cloud computing. Large clouds frequently distribute their services among several sites, each of which is a data center. To accomplish coherence, cloud computing depends on sharing resources. It often uses a "pay-as-you-go" approach, which can assist in reducing capital expenses but may also result in unforeseen running expenses for users.

What is Cloud Computing Reference Model?

The cloud computing reference model is an abstract model that divides a cloud computing environment into abstraction layers and cross-layer functions to characterize and standardize its functions. This reference model divides cloud computing activities and functions into three cross-layer functions and five logical layers.

Each of these layers describes different things that might be present in a cloud computing environment, such as computing systems, networking, storage equipment, virtualization software, security measures, control and management software, and so forth. It also explains the connections between these organizations. The five layers are the Physical layer, virtual layer, control layer, service orchestration layer, and service layer.

Cloud Computing reference model is divided into 3 major service models:

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

2.1.1 Platform as a Service

Platform as a service (PaaS) is a category of cloud computing services that provide a computing platform and a solution stack as a service. Along with software as a service (SaaS) and infrastructure as a service (IaaS), it is a service model of cloud computing. PaaS offerings facilitate the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities.
Technically, a PaaS is an Application Platform comprised of an operating system, middleware and other software that allows applications to run on the cloud with much of the management, security, scaling and other stack related headaches abstracted away. This allows you to focus on two things: customers and developing your application. Let the PaaS deal with system administration details like setting up servers or VMs, installing libraries or frameworks, configuring testing tools, etc.
Platform as a Service allows users to create software applications using tools supplied by the provider. PaaS services can consist of preconfigured features that customers can subscribe to; they can choose to include the features that meet their requirements while discarding those that do not. PaaS works on top of IaaS and will do all of that work automatically.

Characteristics of PaaS

1. Multi-tenant architecture A PaaS offering must be multi-tenanted. A multi-tenant platform is one that uses common computing resources including hardware, operating system, software (i.e. application code), and a single underlying database with a shared schema to support multiple customers simultaneously. This is in direct contrast to the traditional client/server architecture, which requires an entire stack of hardware and software to be dedicated to each tenant (customer).

2. Customizable /Programmable User Interface

PaaS offering should provide the ability to construct highly flexible user interfaces via a simple "drag & drop" methodology that permits the creation and configuration of UI components on the fly. Furthermore, given the growing set of Web devices, additional flexibility to use other technologies such as CSS, AJAX and Adobe Flex to specify the appearance of the application's interface should be available to the UI designer.

3. Unlimited Database Customizations

Database used by application should have option of customization for more flexibility in application development. Specifying relationships between objects, a key requirement of any sophisticated business application, must be possible through the declarative Web-based interface. Other mandatory functions include the ability to incorporate validation rules and permissions at the object/field level and the ability to specify auditing behavior.

4. Automation

PaaS environments automate the process of deploying applications to infrastructure, configuring application components, provisioning and configuring supporting technology like load balancers and databases, and managing system change based on policies set by the user. While IaaS is known for its ability to shift capital costs to operational costs through outsourcing, only PaaS is able to cut down costs across the development, deployment and management aspects of the application lifecycle.

5. Security

The PaaS offering should provide a flexible access control system that allows detailed control over what users of the SaaS application can see and the data each user can access. Definition of access from the application level (including tabs, menus, objects, views, charts, reports and workflow actions) to the individual field level should be possible. Defining an access control model should be possible through the creation of groups and roles and the assignment of users to either groups or roles.

6. Runtime Framework:
This is the "software stack" aspect of PaaS, and perhaps the aspect that comes first to mind for most people. The PaaS runtime framework executes end-user code according to policies set by the application owner and cloud provider. PaaS runtime frameworks come in many flavors,

some based on traditional application runtimes, others based on 4GL and visual programming concepts, and some with pluggable support for multiple application runtimes.

The Traditional On-Premises Model

The traditional approach to developing and running applications on-premises has always been complex, expensive and risky. Producing your own solution never brought any guarantee of success. Each application has been designed to meet their specific requirements within each business. Each solution requires a specific programming hardware, an operating system, a database, often a middle-ware package, mail and web servers, etc. Once environment was created in hardware and software, a team of developers had to navigate a complex programming platform to build their own applications. Additionally, a team of network, database and system management was needed to keep everything in perfect driving conditions. Inevitably, developers were forced to change the application on behalf of a detail of the business, generating new cycles of testing before being distributed.

PaaS model offers a choice of faster and more cost-effective application development and delivery. Furthermore, PaaS provides all the infrastructure needed to run applications over the Internet. Just like Google, iTunes and Youtube, this cloud computing model allows new functionality to be delivered in emerging markets through web browsers. PaaS is based on a model of mediation or subscription, and users only pay for what they use. The PaaS model in its range also includes other facilities such as, application design and development, testing, deployment and hosting as well as integration, security, scalability, storage, status management, control panel, etc.

2.1.2 Software as a Service

Software as a service , sometimes referred to as "on-demand software", is a software delivery model in which software and associated data are centrally hosted on the cloud. SaaS is typically accessed by users using a thin client via a web browser.

In this model, the software is not hosted on the customers' individual computers. Under the SaaS model, a vendor is responsible for the creation, updating, and maintenance of software. Customers buy a subscription to access it, which includes a separate license, or seat, for each person that will use the software.

SaaS has become a common delivery model for many business applications, including accounting, collaboration, customer relationship management (CRM), management information systems(MIS), enterprise resource planning (ERP), invoicing, human resource management (HRM), content management (CM) and service desk management.

The emergence of SaaS as an effective software-delivery mechanism creates an opportunity for IT departments to change their focus from deploying and supporting applications to managing the services that those applications provide.

Unlike traditional software which is conventionally sold as a perpetual license with an up-front cost (and an optional ongoing support fee), SaaS providers generally price applications using a subscription fee, most commonly a monthly fee or an annual fee. Consequently, the initial setup cost for SaaS is typically lower than the equivalent enterprise software. SaaS vendors

typically price their applications based on some usage parameters, such as the number of users ("seats") using the application. However, because in a SaaS environment customers' data reside with the SaaS vendor, opportunities also exist to charge per transaction, event, or other unit of value.

Benefits of SaaS Model:

The SaaS model helps enterprises ensure that all locations are using the correct application version and, therefore, that the format of the data being recorded and conveyed is consistent, compatible, and accurate. By placing the responsibility for an application onto the doorstep of a SaaS provider, enterprises can reduce administration and management burdens they would otherwise have for their own corporate applications. SaaS also helps to increase the availability of applications to global locations. SaaS also ensures that all application transactions are logged for compliance purposes. The benefits of SaaS to the customer are very clear;

- Streamlined administration

- Automated update and patch management services

- Data compatibility across the enterprise (all users have the same version of software)

- Facilitated, enterprise-wide collaboration

- Global accessibility

Save money by not having to purchase servers or other software to support use.

 Focus Budgets on competitive advantage rather than infrastructure.

 Monthly obligation rather than up front capital cost.

Characteristics of SaaS

 Simple and quick system implementation: As Saas sits on top of PaaS and IaaS, deploying any enterprise level software becomes easy and quick as SaaS inherits all the features of underlying infrastructure. Also since SaaS is scalable, any user request can be addressed with required elasticity.

 Transparent and low pricing: With common infrastructure, cloud vendor can leverage their infrastructure with lower cost and transparency. End result of this directly impacts customer cost of software implementation as they gets it cheaper with enhanced security and high availability.

 Multitenant Architecture: A multitenant architecture, in which all users and applications share a single, common infrastructure and code base that is centrally maintained. Because SaaS vendor clients are all on the same infrastructure and code base, vendors can innovate more quickly and save the valuable development time previously spent on maintaining numerous versions of outdated code.

 Easy software maintenance and Customization: The ability for each user to easily customize applications to fit their business processes without affecting the common

infrastructure. Because of the way SaaS is architected, these customizations are unique to each company or user and are always preserved through upgrades. That means SaaS providers can make upgrades more often, with less customer risk and much lower adoption cost.

Better Access: Improved access to data from any networked device while making it easier to manage privileges, monitor data use, and ensure everyone sees the same information at the same time.

2.1.3 Infrastructure-as-a-Service (IaaS):

Infrastructure-as-a-Service (IaaS) is the delivery of computer infrastructure (typically a platform virtualization environment) as a service. IaaS leverages significant technology, services, and data center investments to deliver IT as a service to customers. Unlike traditional outsourcing, which requires extensive due diligence, negotiations and complex, lengthy contract vehicles, IaaS is centered around a model of service delivery that provisions a predefined, standardized infrastructure specifically optimized for the customer's applications.

Infrastructure as a Service (IaaS) **includes the capability to provision processing, storage, networks, and other fundamental computing resources; the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems;** storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls). Services offered by this paradigm include: server hosting, web servers, storage, computing hardware, operating systems, virtual instances, load balancing, Internet access, and bandwidth provisioning. **IaaS clouds often offer additional resources such as; images in a virtual machine image library, raw (block) and file-based storage, firewalls, IP addresses, virtual local area networks (VLANs) and software bundles.** Examples of IaaS providers include **Amazon CloudFormation, Amazon EC2, Windows Azure Virtual Machines, DynDNS, Google Compute Engine, HP Cloud.**

A typical Infrastructure as a Service offering can deliver the following features and benefits:

 Scalability; resource is available as and when the client needs it and, therefore, there are no delays in expanding capacity or the wastage of unused capacity

 No investment in hardware; the underlying physical hardware that supports an IaaS service is set up and maintained by the cloud provider, saving the time and cost of doing so on the client side

 Utility style costing; the service can be accessed on demand and the client only pays for the resource that they actually use

 Location independence; the service can usually be accessed from any location as long as there is an internet connection and the security protocol of the cloud allows it

 Physical security of data center locations; services available through a public cloud, or private clouds hosted externally with the cloud provider, benefit from the physical security afforded to the servers which are hosted within a data center

 No single point of failure; if one server or network switch, for example, were to fail, the broader service would be unaffected due to the remaining multitude of hardware resources and

redundancy configurations. For many services if one entire data center were to go offline, never mind one server, the IaaS service could still run successfully.

Some of the most popular IaaS solutions are discussed as below:

Compute as a service: One of the most ubiquitous(universal) IaaS offerings today, compute as a service provides compute capacity that includes servers, operating system access, firewalls, routers and load balancing on demand.

Web hosting: Many organizations rely on their websites for marketing and revenue, and any glitch (malfunction) in operations can mean a loss of business. **Moving a website to an IaaS based model ensures that the website won't get bogged down during peak traffic times** — and that organizations won't have to overpay for capacity to manage those traffic spikes. What's more, loads will always be balanced, and uptime is guaranteed, thanks to SLAs. Other perks include offsite backup and fast connections for eliminating slow page and content downloads, no matter how much rich media a site includes.

Storage as Service: Storage as a Service is a business model in which a large company rents space in their storage infrastructure to a smaller company or individual. In the enterprise, SaaS vendors are targeting secondary storage applications by promoting SaaS as a convenient way to manage backups. The key advantage to SaaS in the enterprise is in cost savings -- in personnel, in hardware and in physical storage space.

Disaster recovery and backup as a service: The idea behind moving disaster recovery to the cloud is to ensure that organizations have uninterrupted access to data and applications, regardless of emergencies, such as power outages, natural disasters or system failures.

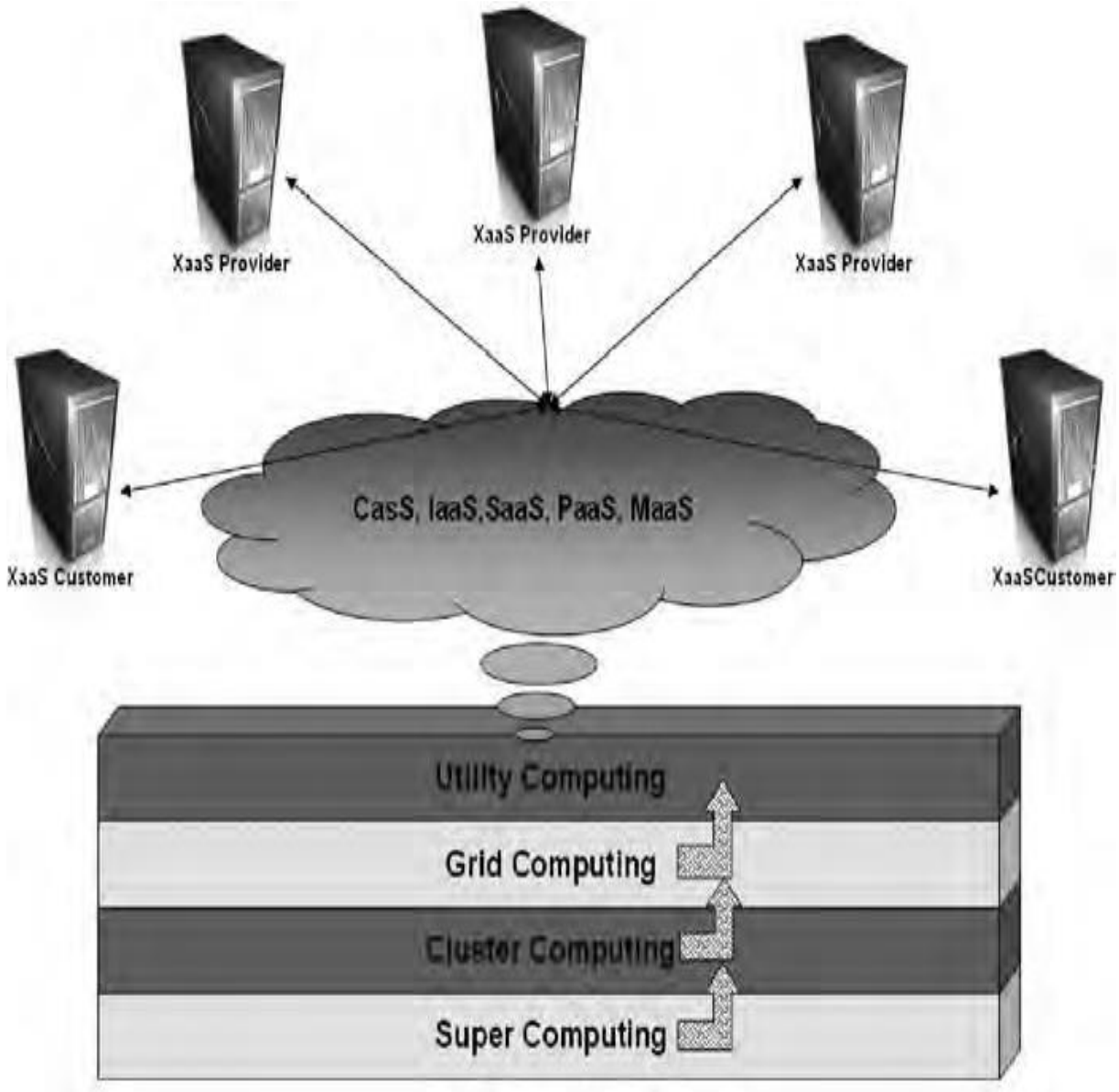
Modern On-Demand Computing: On-demand (OD) computing is an increasingly popular enterprise model in which computing resources are made available to the user as needed. The resources may be maintained within the user's enterprise, or made available by a service provider. The on-demand model evolved to overcome the challenge of being able to meet fluctuating resource demands efficiently. Because demand for computing resources can vary drastically from one time to another, maintaining sufficient resources to meet peak requirements can be costly.
On-demand computing products are rapidly becoming prevalent in the marketplace. Computer Associates, HP, IBM, Microsoft, and Sun Microsystems are among the more prominent on-demand

vendors. These companies refer to their on-demand products and services by a variety of names. Concepts such as grid computing, utility computing, autonomic computing, and adaptive management seem very similar to the concept of on-demand computing.

The major advantage of On Demand Computing (ODC) is low initial cost, as computational resources are essentially rented when they are required. This provides cost savings over purchasing them outright.

Amazon Web Services:

Amazon was the first providers of cloud computing (*http://aws.amazon.com*); it announced a limited public beta release of its Elastic Computing platform called *EC2* in August 2006.



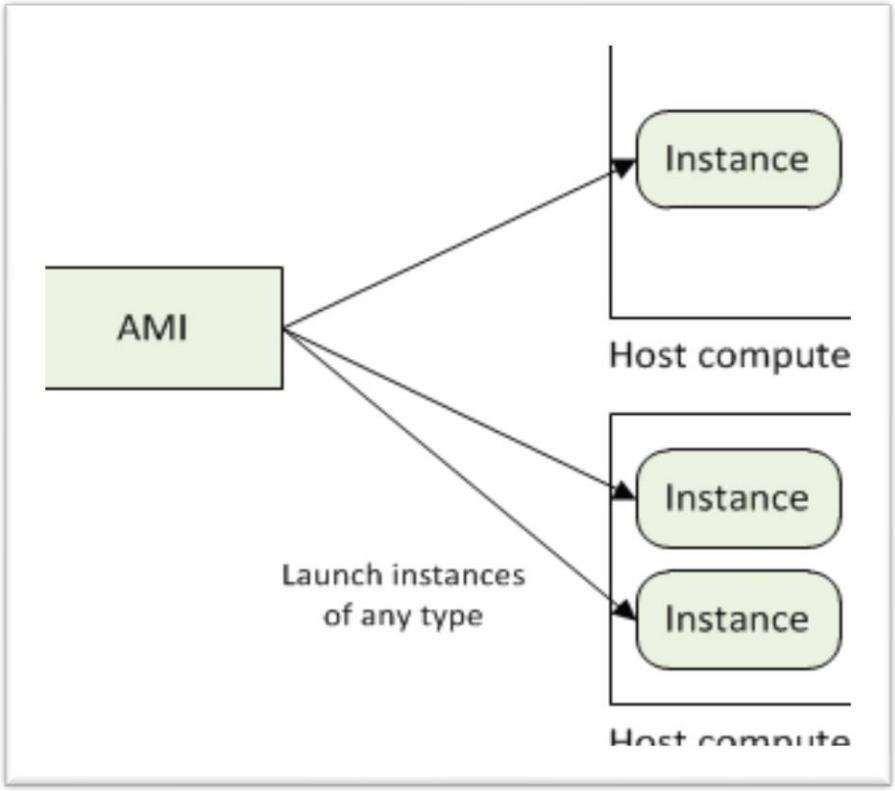
AMAZON EC2 SERVICE:

Elastic Compute Cloud (EC2) **is a Web service with a simple interface for launching instances of an application under several operating systems, such as several Linux**

distributions, Microsoft Windows Server 2003 and 2008, OpenSolaris, FreeBSD, and NetBSD.

Amazon Elastic Compute Cloud (EC2) is a central part of Amazon cloud computing platform Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a Web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic".

It reduces the time required to obtain and boot new server instances to minutes, allowing customers to quickly scale capacity as their computing demands dictate. It changes the economics of computing by allowing clients to pay only for capacity they actually use. It provides developers the tools needed to build failure-resilient applications and isolate themselves from common failure scenarios. To use the EC2, a subscriber creates an Amazon Machine Image (AMI) containing the operating system, application programs and configuration settings. Then the AMI is uploaded to the Amazon Simple Storage Service (Amazon S3) and registered with Amazon EC2, creating a so-called AMI identifier (AMI ID). Once this has been done, the subscriber can requisition virtual machines on an as-needed basis. Capacity can be increased or decreased in real time from as few as one to more than 1000 virtual machines simultaneously. Billing takes place according to the computing and network resources consumed.



The idea of EC2 is to lighten the cost of buying servers to host a system, but more importantly to eliminate the wasted time systems engineers devote to managing hard assets. Instead of buying servers to increase capacity or add new features, you simply buy more gigabytes on EC2. Amazon sells it in subscription form, with subscriptions based on how much capacity you use.

A user can,

i. launch an instance from an existing AMI and terminate an instance;
ii. start and stop an instance;
iii. create a new image;
iv. add tags to identify an image; and reboot an instance.

EC2 is based on the Xen virtualization strategy. **In *EC2* each virtual machine functions as a virtual private server and is called an *instance;* an instance specifies the maximum amount of resources available to an application, the interface for that instance, as well as, the cost per hour. This is a web service that provides resizable computing capacity in the cloud.** It is designed to make

web-scale computing easier for developers and offers many advantages to customers:

- It is a web service interface that allows customers to obtain and configure capacity with minimal effort.
- It provides users with complete control of their (leased) computing resources and lets them run on a proven computing environment.
- It reduces the time required to obtain and boot new server instances to minutes, allowing customers to quickly scale capacity as their computing demands dictate.
- It changes the economics of computing by allowing clients to pay only for capacity they actually use.

  Amazon EC2 presents a true virtual computing environment, allowing clients to use a web-based interface to obtain and manage services needed to launch one or more instances of a variety of operating systems (OSs). Clients can load the OS environments with their customized applications. They can manage their network's access permissions and run as many or as few systems as needed. In order to use Amazon EC2, clients first need to create an Amazon Machine Image (AMI). This image contains the applications, libraries, data, and associated configuration settings used in the virtual computing environment. Amazon EC2 offers the use of preconfigured images built with templates to get up and running immediately. Once users have defined and configured their AMI, they use the Amazon EC2 tools provided for storing the AMI by uploading the AMI into Amazon S3. Amazon S3 is a repository that provides safe, reliable, and fast access to a client AMI. Before clients can use the AMI, they must use the Amazon EC2 web service to configure security and network access.

  A user can interact with *EC2* using a set of SOAP messages, (*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. SOAP uses TCP and more recently UDP transport protocols; it can also be stacked above other application layer protocols such as HTTP, SMTP. 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*), and can list available AMI images, boot an instance from an image, terminate an image, display the running instances of a user, display console output, and so on

Amazon EC2 Service Characteristics:

a) **Dynamic Scalability:** Amazon EC2 enables users to increase or decrease capacity in a few minutes. Users can invoke a single instance, hundreds of instances, or even thousands of instances simultaneously.

- b) **Full Control of Instances:** Users have complete control of their instances. They have root access to each instance and can interact with them as one would with any machine. Instances can be rebooted remotely using web service APIs. Users also have access to console output of their instances. Once users have set up their account and uploaded their AMI to the Amazon S3 service, they just need to boot that instance. It is possible to start an AMI on any number of instances (or any type) by calling the *Run Instances* API that is provided by Amazon.
- c) **Configuration Flexibility:** Configuration settings can vary widely among users. They have the choice of multiple instance types, operating systems, and software packages. Amazon EC2 allows them to select a configuration of memory, CPU, and instance storage that is optimal for their choice of operating system and application. For example, a user's choice of operating systems may also include numerous Linux distributions, Microsoft Windows Server, and even an Open Solaris environment, all running on virtual servers.

- d) **Integration with Other Amazon Web Services:** Amazon EC2 works in conjunction with a variety of other Amazon web services. For example, Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB, Amazon Simple Queue Service (Amazon SQS). **Amazon S3** provides a web services interface that allows users to store and retrieve any amount of data from the Internet at any time, anywhere. It gives developers direct access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure Amazon uses to run its own global network of web sites. **Amazon SimpleDB** is another web-based service, designed for running queries on structured data stored with the Amazon Simple Storage Service (Amazon S3) in real time. . **Amazon Simple Queue Service (Amazon SQS)** is a reliable, scalable, hosted queue for storing messages as they pass between computers.

e) **Reliable and Resilient (strong) Performance Amazon Elastic Block Store (EBS):** is yet another Amazon EC2 feature that provides users powerful features to build failure-resilient applications. Amazon EBS offers persistent storage for Amazon

EC2 instances. Amazon EBS volumes provide "off-instance" storage that persists independently from the life of any instance.

f) Support for Use in Geographically Disparate Locations: Amazon EC2 provides users with the ability to place one or more instances in multiple locations. Amazon EC2 locations are composed of Regions (such as North America and Europe) and Availability Zones. Regions consist of one or more Availability Zones, are geographically dispersed, and are in separate geographic areas or countries. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity.

Examples of Cloud Computing Reference Model Apart From NIST

1. IBM Architecture
2. Oracle Architecture
3. HP Architecture
4. Cisco Reference Architecture

2.2 Cloud Computing Deployment Models (Types):

Public: The infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services. It is also referred to as 'external' Cloud that describes the conventional meaning of Cloud computing: scalable, dynamically provisioned, often virtualized resources available over the internet from an off-site third party provider, which divides up resources and bills its customers on a 'utility' basis.

Public cloud applications, storage, and other resources are made available to the general public by a service provider. These services are free or offered on a pay-per-use model. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure and offer access only via Internet (direct connectivity is not offered).

Private: The infrastructure is operated solely for an organization; it may be managed by the organization or a third party and may exist on or off the premises of the organization. It is also referred to as 'corporate' or 'internal' Cloud, term used to denote a proprietary computing architecture providing hosted services on private networks.

Undertaking a private cloud project requires a significant level and degree of engagement to virtualize the business environment, and it will require the organization to reevaluate decisions about existing resources. When it is done right, it can have a positive impact on a business, but

every one of the steps in the project raises security issues that must be addressed in order to avoid serious vulnerabilities

Community: Community cloud shares infrastructure between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party and hosted internally or externally. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized

A community cloud contains features of the public and private cloud models. Like a public cloud, the community cloud may contain software, data storage, and computing resources that are utilized by multiple organizations. Where this model differs from the public model is that the infrastructure is only utilized by a group of organizations that are known to each other. Similarly to a private cloud, these organizations are responsible for the operation of their own infrastructure. The community cloud model can provide greater cost savings than the private cloud while offering some of its security features. This model is best suited for organizations that share common requirements such as security or legal compliance policies. It can be managed by the member organizations or by a third party provider

Hybrid Cloud: This can be a combination of private and public clouds that support the requirement to retain some data in an organization, and also the need to offer services in the cloud. A company may use internal resources in a private cloud maintain total control over its proprietary data. It can then use a public cloud storage provider for backing up less sensitive information. At the same time, it might share computing resources with other organizations that have similar needs. By combining the advantages of the other models, the hybrid model offers organizations the most flexibility.

This model is also used for handling cloud bursting, which refers to a scenario where the existing private cloud infrastructure is not able to handle load spikes and requires a fallback option to support the load. Hence, the cloud migrates workloads between public and private hosting without any inconvenience to the users.

2.3 Cloud design and implementation using SOA

A cloud has some key characteristics: elasticity, self-service provisioning, standards based interfaces, and pay as you go. This type of functionality has to be engineered into the software. To accomplish this type of engineering requires that the foundation for the cloud be well designed and well architected. What about cloud architecture makes this approach possible? The fact is that the services and structure behind the cloud should be based on a modular architectural approach. A modular, component-based architecture enables flexibility and reuse. **A *Service Oriented Architecture (SOA)* is what lies beneath this flexibility.**

SOA is much more than a technological approach and methodology for creating IT systems. It's also a *business* approach and methodology. Companies have used the principles of SOA to deepen the understanding between the business and IT and to help business adapt to change.

One of the key benefits of a service oriented approach is that software is designed to reflect best practices and business processes instead of making the business operate according to the rigid structure of a technical environment. A service-oriented architecture is essentially a collection of services. A service is, in essence, a function that is well defined, self-contained, and does not depend on the context or state of other services. Services most often reflect logical business activities. Some means of connecting services to each other is needed, so services communicate with each other, have an interface, and are message-oriented. The communication between services may involve simple data passing or may require two or more services coordinating an activity. The services generally communicate using standard protocols, which allows for broad interoperability. SOA encompasses legacy systems and processes, so the effectiveness of existing investments is preserved. New services can be added or created without affecting existing services. Service-oriented architectures are not new. **The first service-oriented architectures are usually considered to be the Distributed Component Object Model (DCOM) or Object Request Brokers (ORBs), which were based on the Common Object Requesting Broker Architecture (CORBA) specification.** The introduction of SOA provides a platform for technology and business units to meet business requirements of the modern enterprise. With SOA, your organization can use existing application systems to a greater extent and may respond faster to change requests.

These benefits are attributed to several critical elements of SOA:

1. Free-standing, independent components

2. Combined by loose coupling

3. Message (XML)-based instead of API-based

4. Physical location, etc., not important

For example, a core banking application provides a Fund Transfer service, then the other banking applications such as Treasury, Payment Gateway, ATM Switching, and so on can call or invoke Fund Transfer service without need to worry about where the Fund Transfer is located in the network. This contrasts with the Tight Coupling approach. Each application defining their own Fund Transfer, the problem come when the Transfer Fund logic is change. It will be difficult and require high cost (and time, of course) to set the new logic into each application.

Combining Cloud and SOA:

Cloud services benefit the business by taking the best practices and business process focus of SOA to the next level. These benefits apply to both cloud service providers and cloud service users. Cloud service providers need to architect solutions by using a service-oriented approach to deliver services with the expected levels of elasticity and scalability. Companies that architect and govern business processes with reusable service-oriented components can more easily identify which components can be successfully moved to public and private clouds. A *service oriented architecture (SOA)* is a software architecture for building business applications that implement business processes or services through a set of loosely coupled, black-box components orchestrated to deliver a well defined level of service. This approach lets companies leverage existing assets and create new business services that are consistent, controlled, more easily changed, and more easily managed. SOA is a business approach to designing efficient IT systems that support reuse and give the businesses the flexibility to react quickly to opportunities and threats.

Characterizing SOA :

The principal characteristics of SOA are described in more detail here:

- **SOA is a black-box component architecture.** The *black box* lets you reuse existing business applications; it simply adds a fairly simple adapter to them. You don't need to know every detail of what's inside each component; SOA hides the complexity whenever possible.

- **SOA components are loosely coupled.** Software components are *loosely coupled* if they're designed to interact in a standardized way that minimizes dependencies. One loosely coupled component passes data to another component and makes a request; the second component carries out the request and, if necessary, passes data back to the first. Each component offers a small range of simple services to other components. A set of loosely coupled components does the same work that software components in tightly structured applications used to do, but with loose coupling you can combine and recombine the components in a bunch of ways. This makes a world of difference in the ability to make changes easily, accurately, and quickly.

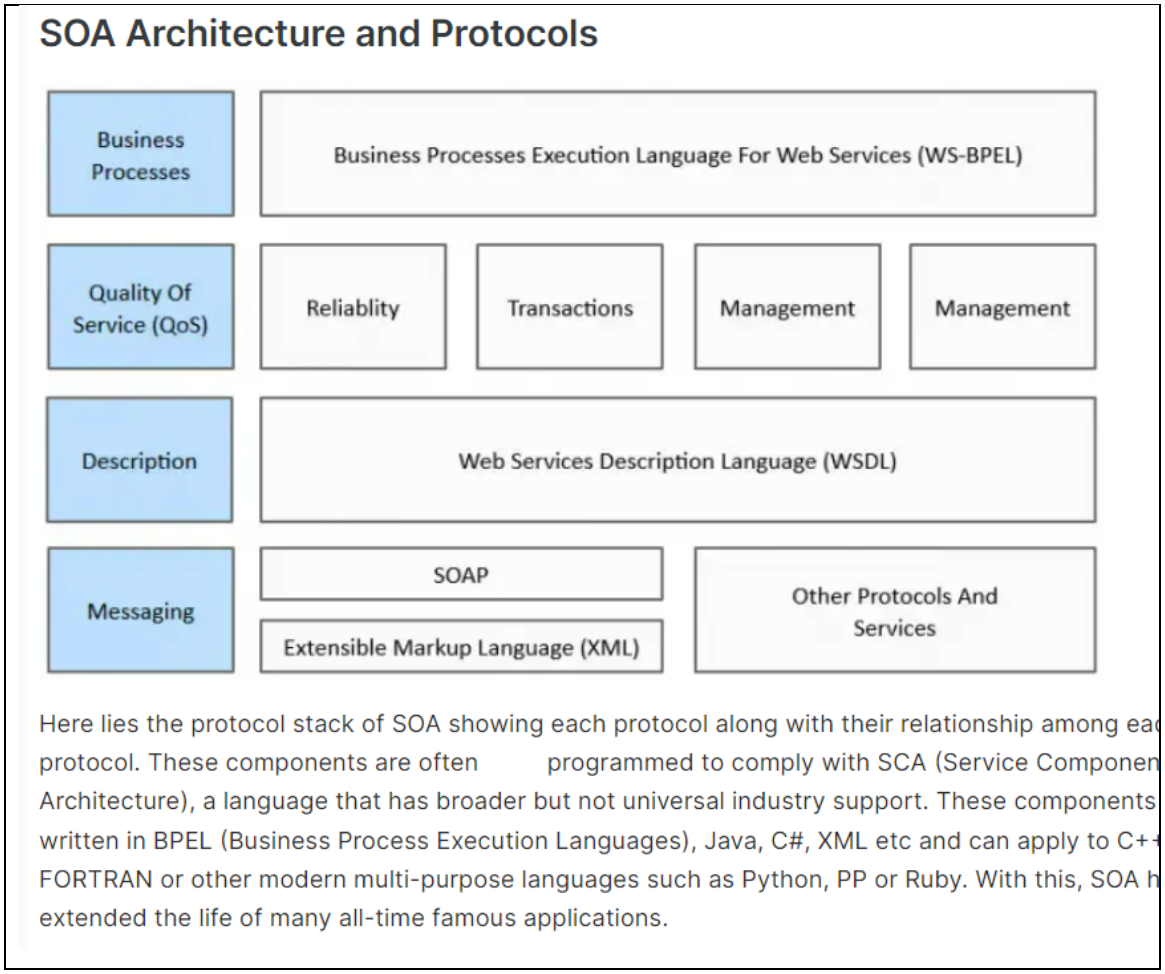
- **SOA components are orchestrated to link through business processes to deliver a well-defined level of service.** SOA creates a simple arrangement of components that, together, deliver a very complex business service. Simultaneously, SOA must provide acceptable service levels. To that end, the components ensure a dependable service level. Service level is tied directly to the best practices of conducting business, commonly referred to as *business process management (BPM)* — BPM focuses on effective design of business process and SOA allows IT to align with business processes.

Some more points related to characteristics.

- In SOA, Services should be **independent** of other services. Altering a service should not affect calling service.
- Services should be **self-contained**. When we talk about a Register Customer service it means, service will do all the necessary work for us, we are not required to care about anything.
- Services should be able to **define themselves**. Services should be able to answer a question what is does? It should be able to tell client what all operations it does, what all data types it uses and what kind of responses it will return.
- Services should be **published** into a location (directory) where anyone can search for it.
- As said, SOA comprises of collection services which communicate via **standard Messages**. Standard messages make them platform independent. (Here standard doesn't mean standard across Microsoft it means across all programming languages and technologies.)
- Services should be able to communicate with each other **asynchronously**.
- Services should support **reliable messaging**.Means there should be a guarantee that request will be reached to correct destination and correct response will be obtained.

Services should support **secure communication**

-



2.4 Security, trust and privacy :

Privacy and trust are both complex notions for which there is no standard, universally accepted definition. Consequently, the relationship between privacy, security and trust is necessarily intricate.

Privacy :

At the broadest level and subsequently in the European Convention on Human Rights and national constitutions and charters of rights such as the UK Human Rights Act 1998. Since at least the 1970s the primary focus of privacy has been personal information, and particularly concerned with protecting individuals from government surveillance and databases, potential mandatory disclosure of privacy databases. A decade later concerns were raised related to direct marketing and telemarketing and, later still, consideration was given to the increasing threat of online identity theft. There are various forms of privacy, ranging from 'the right to be left alone 'control of information about ourselves, 'the rights and obligations of individuals and organizations with respect to the collection, use, disclosure, and retention of personally identifiable information.

In the commercial, consumer context, privacy entails the protection and appropriate use of the personal information of customers, and the meeting of expectations of customers about its use. For organisations, privacy entails the application of laws, policies, standards and processes by which personal information is managed. What is appropriate will depend on the applicable laws, individuals' expectations about the collection, use and disclosure of their personal information and other contextual information, hence one way of thinking about privacy is just as 'the appropriate use

of personal information under the circumstances. Data protection is the management of personal information, and is often used within the European Union in relation to privacy-related laws and regulations (although in US the usage of this term is focussed more on security).

Some personal data elements are considered more sensitive than others, although the definition of what is considered sensitive personal information may vary depending upon jurisdiction and even on particular regulations. In Europe, sensitive personal information is called special categories of data, which refers to information on religion or race, political opinions, health, sexual orientation, trade-union membership and data relating to offences or criminal convictions, and its handling is specially regulated.

Key privacy terminology includes the notion of data controller, data processor and data subject. Their meaning is as follows:

Data controller: An entity (whether a natural or legal person, public authority, agency or other body) which alone, jointly or in common with others determines the purposes for which and the manner in which any item of personal information is processed

Data processor: An entity (whether a natural or legal person, public authority, agency or any other body) which processes personal information on behalf and upon instructions of the Data Controller

Data subject: An identified or identifiable individual to whom personal information relates, whether such identification is direct or indirect (for example, by reference to an identification number or to one or more factors specific to physical, physiological, mental, economic, cultural or social identity).

The fair information practices developed in US in 1970s and later adopted and declared as principles by the Organisation for Economic Cooperation and Development (OECD) and the Council of Europe form the basis for most data protection and privacy laws around the world.

These principles can be broadly described as follows:

1. Data collection limitation: data should be collected legally with the

consent of the data subject where appropriate and should be limited

to the data that is needed.

2. Data quality: data should be relevant and kept accurate.

3. Purpose specification: the purpose should be stated at the time of data collection.

4. Use limitation: personal data should not be used for other purposes unless with the consent of the individual.

5. Security: personal data should be protected by a reasonable degree of security.

6. Openness: individuals should be able to find out what personal data is held and how it is used by an organization.

7. Individual participation: an individual should be able to obtain details of all information about them held by a data controller and challenge it if incorrect.

Security

“Preservation of confidentiality, integrity and availability of information; in addition, other properties such as authenticity, accountability, non-repudiation and reliability can also be involved.”

Confidentiality is commonly but erroneously equated with privacy by some security practitioners and is:

“The property that information is not made available or disclosed to unauthorized individuals, entities or processes”

Note that security is actually one of the core privacy principles, as considered in the previous subsection. Correspondingly, it is a common requirement under the law that if a company outsources the handling of personal information or confidential data to another company, it has some responsibility to make sure the outsourcer uses “reasonable security” to protect those data.

Specifically, to ensure the security of the processing of such information, data controllers must implement appropriate technical and organizational measures to protect it against:

• **Unauthorised access or disclosure:** in particular where the processing involves the transmission of data over a network

• **Destruction:** accidental or unlawful destruction or loss

• **Modification:** inappropriate alteration

• **Unauthorised use:** all other unlawful forms of processing

Privacy differs from security, in that it relates to handling mechanisms for personal information, dealing with individual rights and aspects like fairness of use, notice, choice, access, accountability and security. Many privacy laws also restrict the transborder data flow of personal information.

Security mechanisms, on the other hand, focus on provision of protection mechanisms that include authentication, access controls, availability, confidentiality, integrity, retention, storage, backup, incident response and recovery. Privacy relates to personal information only, whereas security and confidentiality can relate to all information.

Trust

Trust is a complex concept for which there is no universally accepted scholarly definition. Evidence from a contemporary, cross-disciplinary collection of scholarly writing suggests that a widely held definition of trust is as follows:

“Trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behaviour of another”

Trust is a broader notion than security as it includes subjective criteria and experience. Correspondingly, there exist both hard (security-oriented) and soft trust (i. e. non-security oriented trust) solutions . “Hard” trust involves aspects like authenticity, encryption, and security in transactions, whereas “soft” trust involves human psychology, brand loyalty, and userfriendliness. Some soft issues are involved in security, nevertheless. An example of soft trust is reputation, which is a component of online trust that is perhaps a company’s most valuable asset (although of course a

CSP's reputation may not be justified). Brand image is associated with trust and suffers if there is a breach of trust or privacy.

People often find it harder to trust on-line services than off-line services , often because in the digital world there is an absence of physical cues and there may not be established centralized authorities. The distrust of on-line services can even negatively affect the level of trust accorded to organizations that may have been long respected as trustworthy.

There are many different ways in which on-line trust can be established: security may be one of these (although security, on its own, does not necessarily imply trust. Some would argue that security is not even a component of trust: Nissenbaum argues that the level of security does not affect trust . On the other hand, an example of increasing security to increase trust comes from people being more willing to engage in ecommerce if they are assured that their credit card numbers and personal data are cryptographically protected .