

## SCHOOL OF COMPUTING

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**UNIT – II – Fog and Cloud computing – SITA1503**

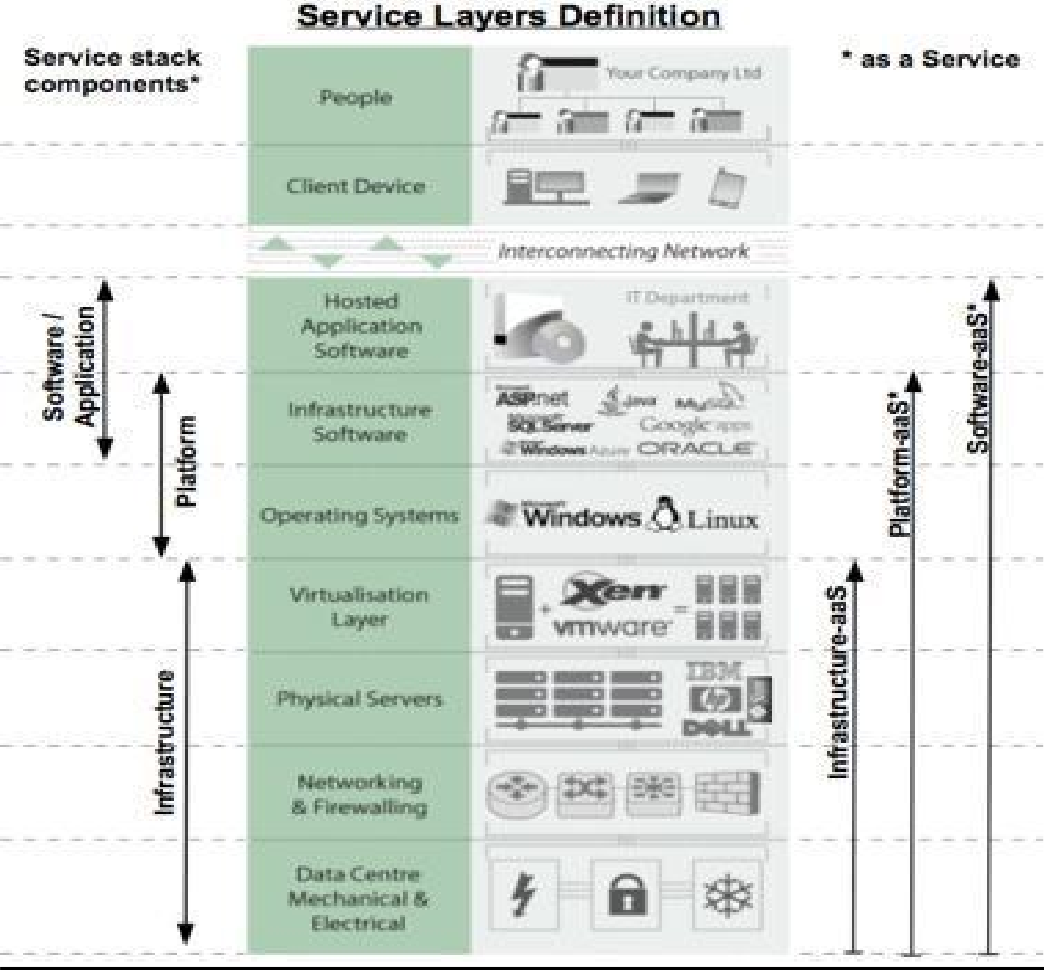
**UNIT 2**

# Cloud Service Models

**Topics:**

Software as a Service (SaaS) - Infrastructure as a Service (IaaS)- Platform as a Service (PaaS)- Web services – Service Oriented Architecture (SoA) - Elastic Computing - On Demand Computing- Service Management in Cloud Computing - Multi-tenancy computing architecture.

# Cloud Services



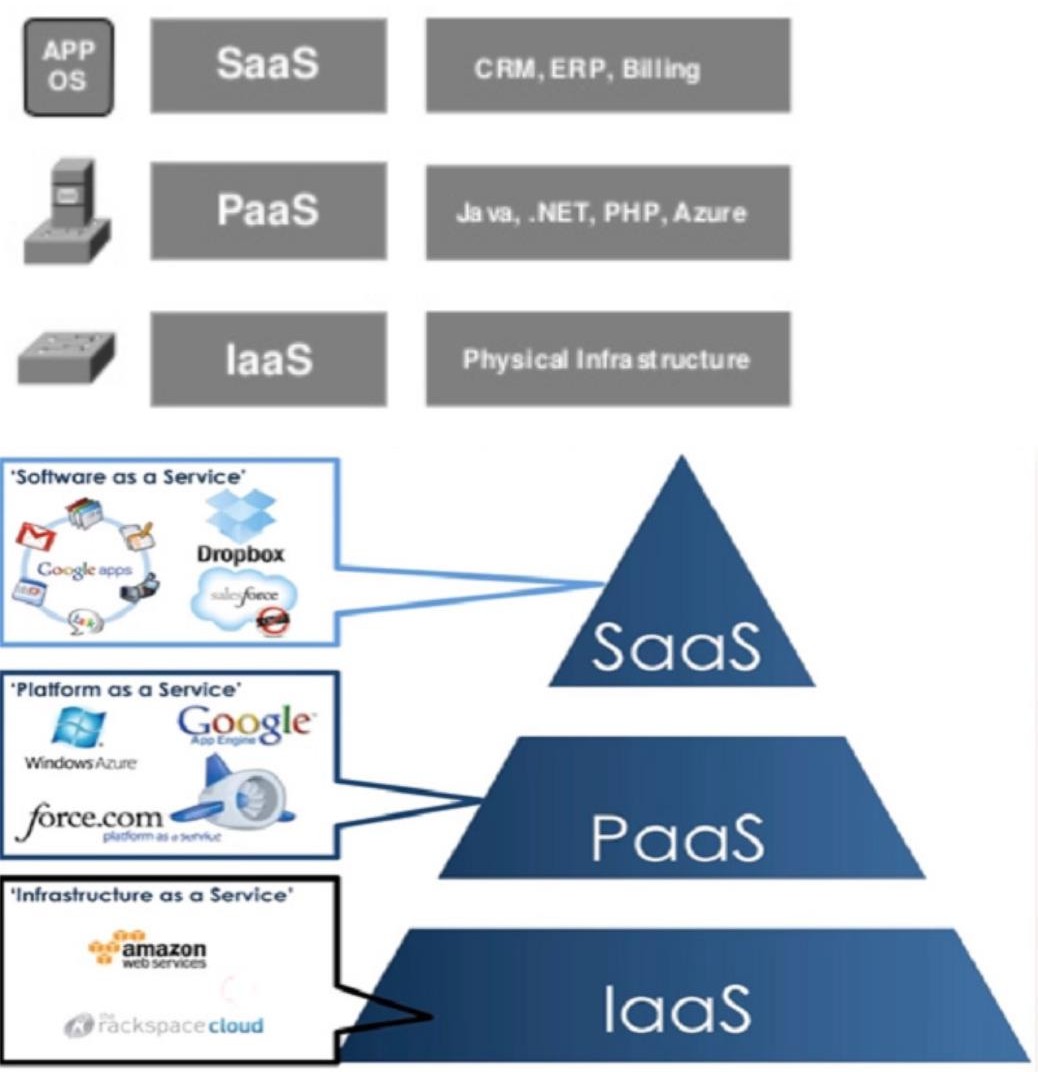
**Fig 2.1 Cloud Services**

# 2.1 Software as a Service



**Fig 2.2 Saas**

* + - SaaS (software-as-a-service). WAN-enabled application services (e.g., Google
    - Software as a Service (SaaS) This is a public cloud service model where the application is 100% managed by the cloud provider.
    - SaaS removes the need for organizations to install and run applications on their own computers or in their own data centers.
    - This eliminates the expense of hardware acquisition, provisioning and maintenance, as well as software licensing, installation and support.
    - Software-as-a-Service (SaaS) has evolved from limited on-line software delivery in 1990s to a fully matured “direct-sourcing” business model for enterprise applications.
    - SaaS is one of the fastest growing concepts: more than 10 million companies will be using SaaS in the next 5 - 10 years; more than 50% of all Fortune 500 companies are already using SaaS.
    - According to influential IT institutes, SaaS is the leading business model of choice for 2008/2009
    - Virtually all big software/service vendors (IBM, Microsoft, Oracle, Cisco) are investing heavily in SaaS
    - With the continuously increasing bandwidth and reliability of the internet, using web services over the (public) internet has become a viable option.
    - Microsoft Office 365 is available with the Azure cloud platform.



**Fig 2.3 The SPI model**

The architecture uses an application instance instead of server instances. There is no actual migration of company servers to the cloud. The SaaS model provides single-tenant and multiple tenant services.

The single-tenant dedicates the application instance to the assigned tenant. The multiple tenant application is shared by multiple tenants. The company can manage the security and storage with the single-tenant model. The SaaS application is well suited to internet connectivity. The employees along with their partners and customers can access the application with a variety of network access devices. The SaaS billing model is based on either per usage or monthly subscription. The security compliance requirements for some applications prevented deployment to the SaaS cloud.

Some SaaS providers offer Virtual Private SaaS (VPS) services for additional application security. It is a hybrid deployment model that allows peering with an enterprise or VPC database server.

The peering is for storage purposes only and used for security compliance. Salesforce.com is a leading SaaS provider with a CRM application to customers.

## Benefits of SaaS

* + Flexible payments
  + Scalable usage
  + Automatic updates
  + Accessibility and persistence
  + On Demand Computing

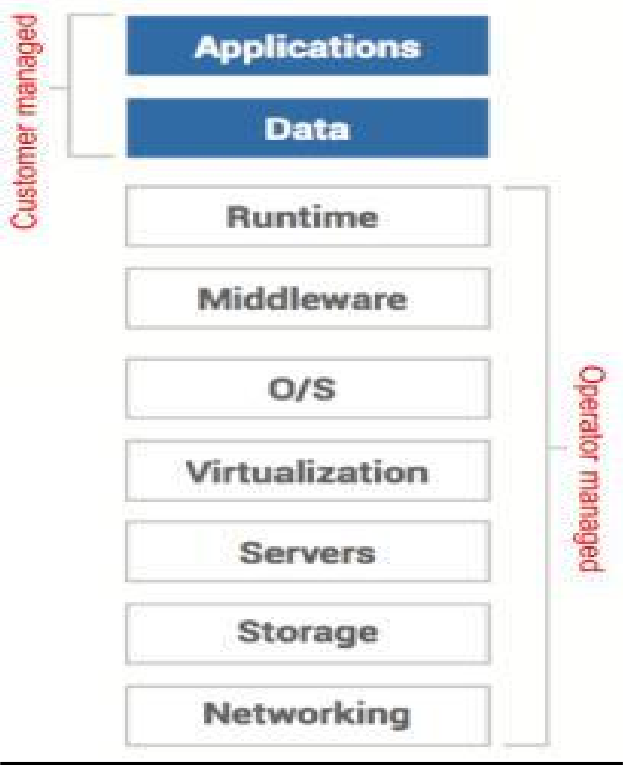
## Opportunities of SaaS

* + Software provided as a service by a software vendor to multiple customers with the following main characteristics:
* Standardization of software
* Service including maintenance, support and upgrades
  + Web based – usage over the (public) internet
  + SaaS offers potential for lowering the Total Cost of Ownership
  + Lower operational costs
  + No large scale, costly, high risk implementations of applications
  + Need few operational resources for application management
  + No platform and hardware (maintenance) costs for application servers
  + Reduced operational complexity: software delivered as a transparent service through the web
  + Minimized software development costs – No lengthy software development and testing cycles
  + Lower costs for software use
  + No software license and annual maintenance fees
  + No expensive software upgrades
  + Lower application consultancy and support costs
  + SaaS allows corporations to focus on core business activities and responsibilities
  + Transparent overview and usage of electronic data and information
  + Automation of iterative, manual tasks
  + Faster Time to Market – easy to scale software
  + More flexibility in changing and modifying application services for business needs – Full scale integration of business processes
  + Control over IT
  + Minimized IT Service Management efforts mainly focused on availability – Well-defined SLAs between the corporation and the IT vendor
  + More predictable cash flow – easier licensing based on access/usage of software
  + Increased productivity and improved user satisfaction
  + Automatic software upgrades with minimal outage

## Limitations

Businesses must rely on outside vendors to provide the software, keep that software up and running, track and report accurate billing and facilitate a secure environment for the business' data.

# 2.2 Platform as a Service



**Fig 2.4 PaaS**

1. The Platform as a Service (PaaS) is a way to rent hardware, operating systems, storage and network capacity over the Internet.
2. PaaS services are,
   * Data services
   * Application runtime
   * Messaging & queuing
   * Application management.

* The PaaS is a computing platform that abstracts the infrastructure, OS, and middle- ware to drive developer productivity.
* The PaaS is foundational elements to develop new applications
* E.g., Google Application Engine, Microsoft Azure, Coghead.

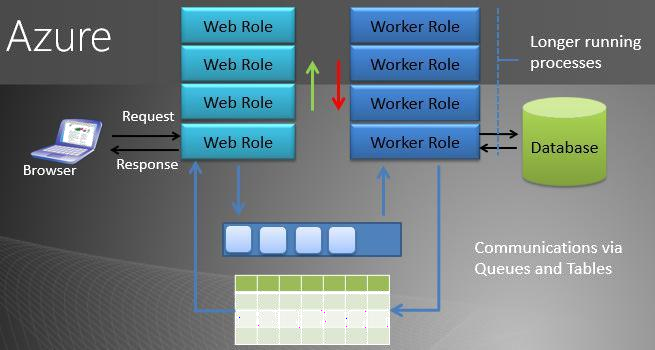
-

## Microsoft Azure

Pay per role instance

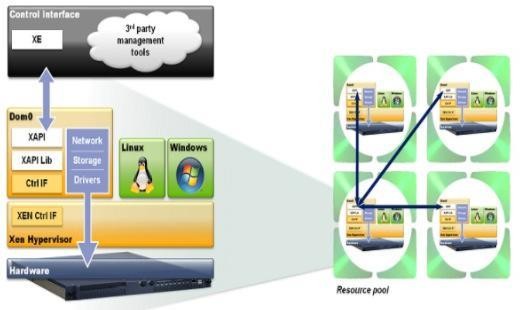
Add and remove instances based on demand

* + Elastic computing!
  + Load balancing is part of the Azure fabric and automatically allocated



**Fig 2.5 Microsof Azure’s Plataform as a Service**

* The PaaS is the delivery of a computing platform and solution stack as a service
* **The Solution stack** is integrated set of software that provides everything a developer needs to build an application for both software development and runtime.



**Fig 2.6 Solution stack**

## -PaaS offers the following

Facilities for application design Application development Application testing, deployment Application services are,

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

All these services may be provisioned as an integrated solution over the web

## Properties and characteristics of PaaS

Scalability Availability Manageability Performance Accessibility

## PaaS Features

It delivers the computing platform as service

The capacities to abstract and control all the underlying resources It helps to providers any smallest unit of resources

To provide a reliable environment for running applications and services Act as a bridge between consumer and hardware

Do not need to care about how to build, configure, manage and maintain the backend environment

It provides a development and testing platform for running developed applications Reduce the responsibility of managing the development and runtime environment

## Advantages of PaaS

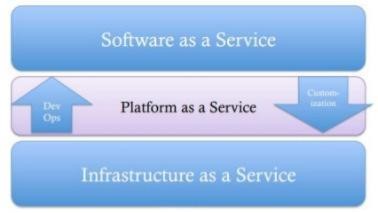
It helps to provide deployment of application without the cost and complexity of buying and managing the hardware and software

It provides all the required to support the complete life cycle of building and delivering web applications and services entirely available from the internet

**Disadvantages of PaaS** Less flexible than IaaS Dependency on provider

Adoption of software / system architecture required

## Evolving from different standards



**Fig 2.7 SPI evolving standard**

Evolving “upwards” from IaaS

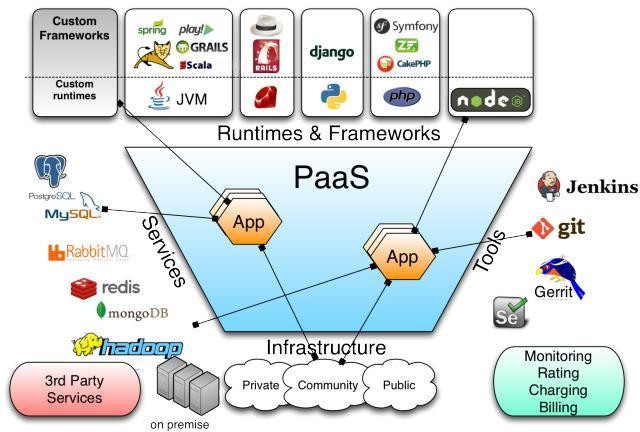
Amazon (Mail, Notification, Events, Databases, Workflow, etc.) Evolving “downwards” from SaaS

Force.com – a place to host additional per-tenant logic. Google App Engine

Evolving “sideways” from middleware platforms

WSO2, Tibco, vmWare, Oracle, IBM

**Generic PaaS Model**



**Fig 2.8 General Paas Model**

# 2.3 Infrastructure as a Service

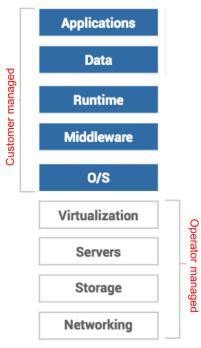
This service offers the computing architecture and infrastructure i.e. all types of computing resources. All resources are offered in a virtual environment, so that multiple users can access it.



**Fig 2.9 IaaS and virtualization**

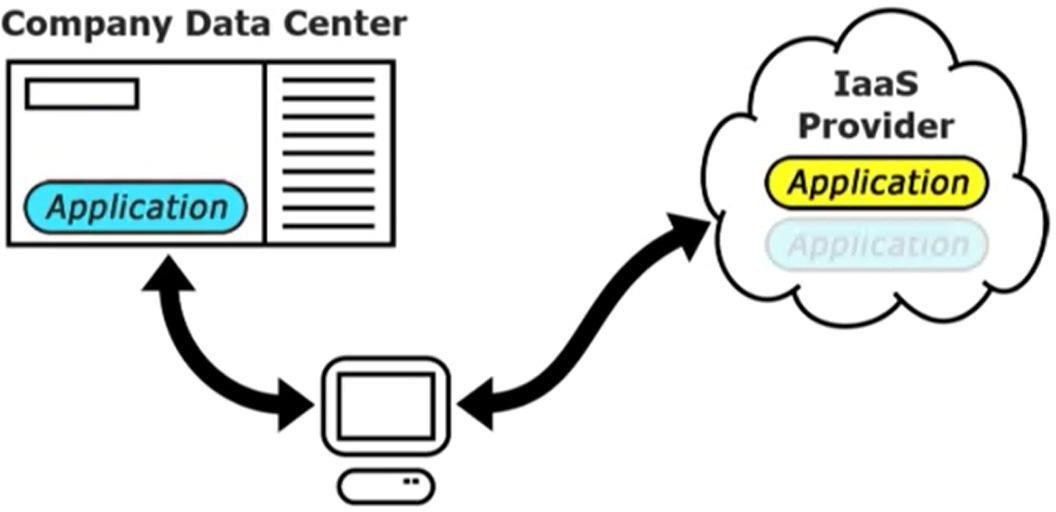
The resources are including, Data storage Virtualization Servers

Networking

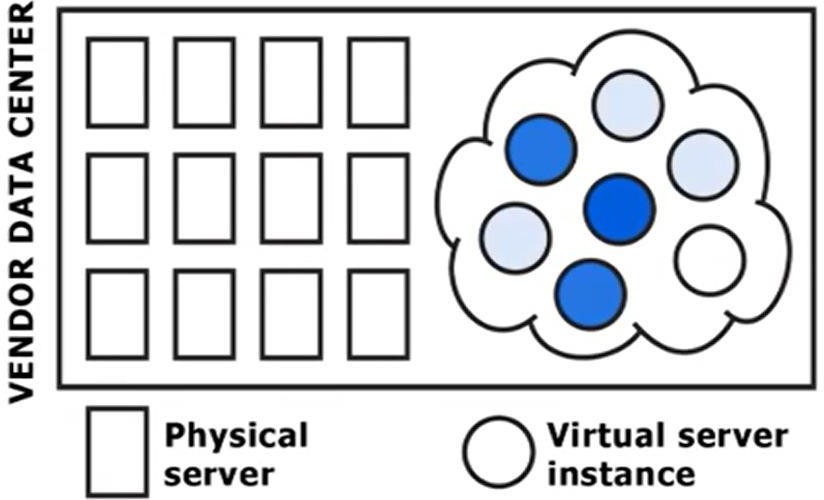


**Fig 2.10 IaaS architecture level**

* + The vendors are responsible for managing all the computing resources which they provided.
  + It allows existing applications to be run on a supplier’s hardware.



## Fig 2.11 User Task in IaaS Cloud



**Fig 2.12 Multiple user can access Virtual instances**

* + The user responsible for handling other resources such as, Applications



Data

Runtime Middleware

## Example IaaS service providers

* + - AWS EC2 / S3 / RDS
    - GoGrid
    - RackSpace

## Pros

The cloud provides the infrastructure

Enhanced scalability i.e. dynamic workloads are supported It is flexible

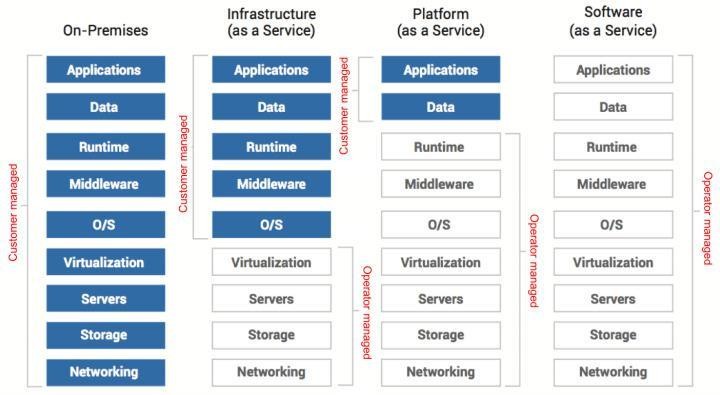
## Cons

Security issues

Network and service delay

## Comparison of cloud services

Blue indicates the levels owned and operated by the organization / Customer White levels are run and operated by the service provider / Operator



**Fig 2.13 Comparison of cloud services**

## Cloud Computing Services Pros

1. Lower computer costs
2. Improved performance:
3. Reduced software costs
4. Instant software updates
5. Improved document format compatibility
6. Unlimited storage capacity
7. Increased data reliability
8. Universal document access
9. Latest version availability
10. Easier group collaboration
11. Device independence

## Cons

* Requires a constant Internet connection
* Does not work well with low-speed connections
* Features might be limited
* Can be slow
* Stored data can be lost
* Stored data might not be secure

# 2.4 Web Services

Web services are XML-centered data exchange systems that use the internet for A2A (application-to-application) communication and interfacing. These processes involve programs, messages, documents, and/or objects.

## Functions of Web Services:

* Available over the internet or intranet networks
* Standardized XML messaging system
* Independent of a single operating system or programming language
* Self-describing via standard XML language
* Discoverable through a simple location method Types of Web Services:

**XML-RPC** (Remote Procedure Call) is the most basic XML protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communication other information from client to server.

**UDDI** (Universal Description, Discovery, and Integration) is an XML-based standard for detailing, publishing, and discovering web services. It’s basically an internet registry for businesses around the world. The goal is to streamline digital transactions and e-commerce among company systems.

**SOAP**, is an XML-based Web service protocol to exchange data and documents over HTTP or SMTP (Simple Mail Transfer Protocol). It allows independent processes operating on disparate systems to communicate using XML.

**REST**provides communication and connectivity between devices and the internet for API-based tasks. Most RESTful services use HTTP as the supporting protocol.

## Web services which are using markup languages:

* + Web template
  + JSON-RPC
  + JSON-WSP
  + Web Services Description Language (WSDL)
  + Web Services Conversation Language (WSCL)
  + Web Services Flow Language (WSFL)
  + Web Services Metadata Exchange (WS-MetadataExchange)
  + XML Interface for Network Services (XINS)

## WSDL

* + WSDL stands for Web Services Description Language
  + WSDL is used to describe web services
  + WSDL is written in XML
  + WSDL is a W3C recommendation from 26. June 2007

|  |  |
| --- | --- |
| **WSDL Element Type** | **Description** |
| <types> | Defines the (XML Schema) data types use by the web service |
| <message> | Defines the data elements for eac operation |
| <portType> | Describes the operations that can b performed and the messages involved. |
| <binding> | Defines the protocol and data format f each port type |

## UDDI:

UDDI is an XML-based standard for describing, publishing, and finding web services.

* UDDI stands for **Universal Description, Discovery, and Integration.**
* UDDI is a specification for a distributed registry of web services.
* UDDI is a platform-independent, open framework.
* UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
* UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services.
* UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
* UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

UDDI has two sections −

* A registry of all web service's metadata, including a pointer to the WSDL description of a service.
* A set of WSDL port type definitions for manipulating and searching that registry.

# 2.5 Service Oriented Architecture

## Service

A service is a program we interact with via message exchanges A system is a set of deployed services cooperating in a given task

## Architecture

It serves as the blueprint for the system Team structure

Documentation organization Work breakdown structure Scheduling, planning, budgeting Unit testing, integration

Architecture establishes the communication and coordination mechanisms among components

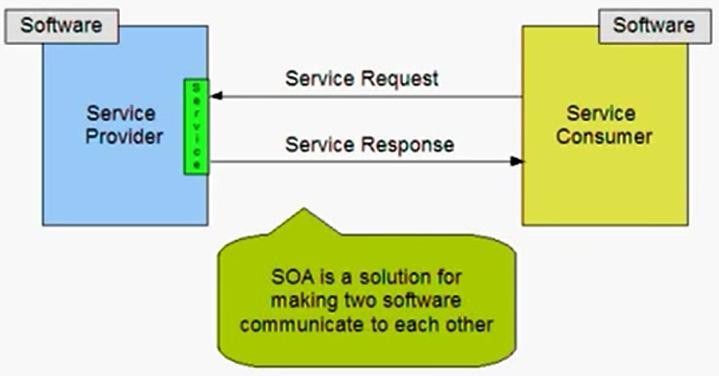
## Software Architecture

It is collection of the fundamental decisions about a software product/solution designed to meet the project's quality attributes (i.e. requirements).

The architecture includes the main components, their main attributes, and their collaboration (i.e. interactions and behavior) to meet the quality attributes.

Architecture usually should be expressed in several levels of

abstraction (depending on the project's size). Architecture is communicated from multiple viewpoints



**Fig 2.14 SOA**

## Why SOA?



**Fig 2.15 Need for SoA**

* + **SOA**

SOA stands for Service Oriented Architecture

It is a design pattern or software architecture which provides application functionality as a service to other applications.

The basic principles of service-oriented architecture are independent of vendors, products and technologies.

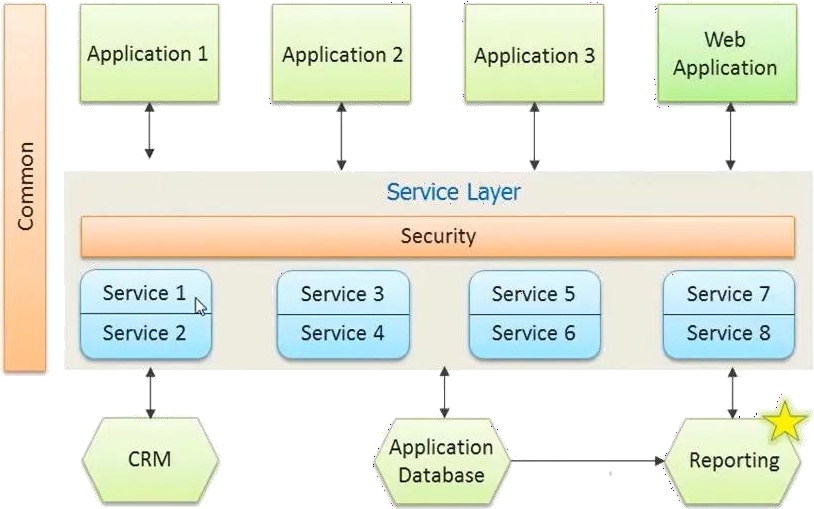
The services are provided to the other components through a communication protocol over a network.

Every service has its own business logic

## SOA Architecture

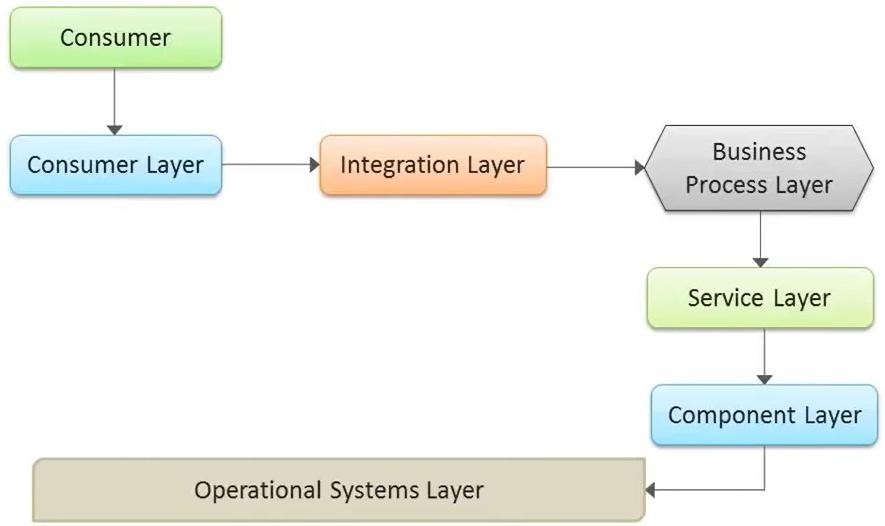
Consumer interface layer – this layer is used by the customer Business process layer – it provides the business process flow

Service layer – this layer comprises of all the services in the enterprises Component layer – this layer has the actual service to be provided Operational system layer – this layer contains the data model



**Fig 2.16 Detailed SoA Architecture**

## SOA Architecture



**Fig 2.17 SoA Architecture**

**SOA – Architecture in details Principles of SOA**

Service loose coupling – service does not have high dependency implementation from outside world

Service reusability – services can be used again and again instead of rewriting them

Service statelessness – they usually do not maintain the state to reduce the resource consumption

Service discoverability – services are registered in registry, so that the client can discover them in the service registry.

## Applications

Manufacturing – E.g. Inventory management

Insurance – Take up the insurance of the employees in companies

## Companies using SOA

Banking Sector

ICICI Bank HDFC Bank

UTI Bank etc

Manufacturing Sector

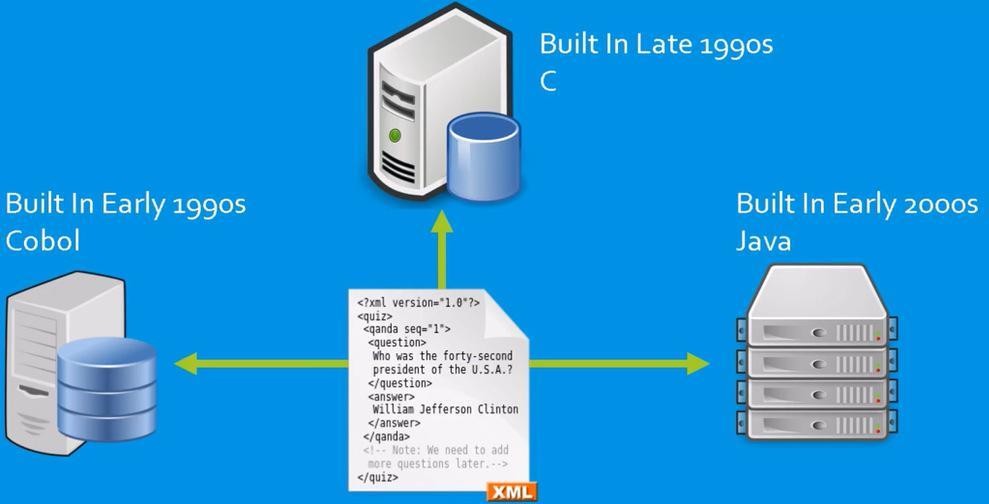
Apollo Tyres Maruthi Hyundai

## Advantages

Interoperability

Programs to run different vendors / locations To interact with different networks

Different operating systems Solution: XML



**Fig 2.18 Evolvement of SoA**

## Scalability

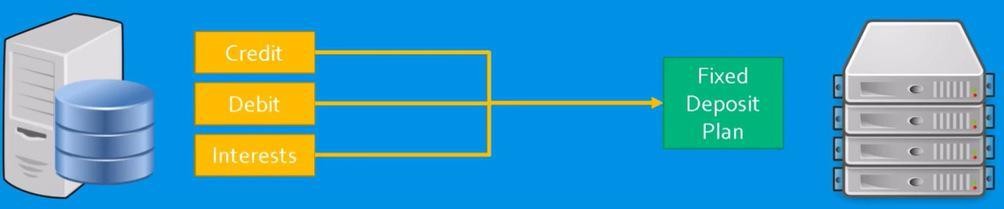
* To extend the processing power of the servers



**Fig 2.19 Scalability**

## Reusability

* If any new systems are introduced, no need to create a new service for every time.



**Fig 2.20 Reusability**

* Parallel application development
* Modular approach
* Easy maintenance
* Greater Reliability
* Improved Software Quality
* Platform Independence
* Increased Productivity

## Disadvantages

* Stand alone, non-distributed applications
* Homogenous application environments
* GUI based applications
* Short lived applications
* Real time applications
* One-way asynchronous communication applications

# 2.6 Elastic Compute Cloud (EC2)

To access the **Elastic Compute Cloud (EC2)** functionality, access the **Amazon EC2** table in the **AWS Console**.

**Elastic Compute Cloud (EC2)** is the engine room of AWS. This is where our servers will operate and run on a day-to-day basis. However, the 'elastic' in EC2 is there for a reason. EC2 is much more than just a bunch of servers! EC2 provides 'resizable compute capacity', or in other words can scale tremendously depending on our capacity requirements at a particular point in time.

EC2 provides the ability to start and stop multiple servers from a single server image, as well as modifying the number of these instances dynamically.

However, there have been some significant differences in the past on how this is implemented, which requires some up-front planning on how we will use the EC2 environment.

## General roles of EC2 in the architecture

EC2 is the backbone of the architecture where our servers are implemented. EC2 will not only run our servers but will manage the capacity that they produce.

## Using EC2

To start using EC2 we must start with an EC2 '**bundle'** or **Amazon Machine Image (AMI).** Both Amazon and third parties such as Right Scale and IBM provide images. For this project, we will be using the default Windows Server Basic AMIs provided by AWS.

Each AMI is a starting point for our instance. Once we have started our Windows instance, we may need to wait up to 15 minutes for AWS to generate our password, so be patient, before we can log on using **Remote Desktop Protocol (RDP)**.

Once our instance has started and we have RDP'd to it, we now have access to install any software that we need onto this instance. But beware, if we fail to create another bundle from our running instance—which we can use to start it next time—then all of our changes will be lost

This is the major difference between standard instances in EC2 and the servers, which we have been familiar with up to now. When installing software onto a server that exists in our own server room, the software tends to remain installed. If we install software on an Amazon EC2 instance, our software (and data) will disappear when our instance is '**terminated**'.

However, recently Amazon has introduced the concept of persistent EC2 images. These are AMIs, which are created on **Elastic Block Store (EBS)** disk. In this specific instance, changes made to the image are persisted when we '**stop**' the image. However, if we terminate the image, the changes are lost.

# 2.7 On Demand Computing

On-demand computing packages computer resources (processing, storage, and so forth) as a metered service similar to that of a public utility. In this model, customers pay for as much or as little processing and storage as they need. Companies that have large demand peaks followed by much lower normal usage periods particularly benefit from utility computing. The company pays more for their peak usage, of course, but their bills rapidly decline when the peak ends and normal usage patterns resume.

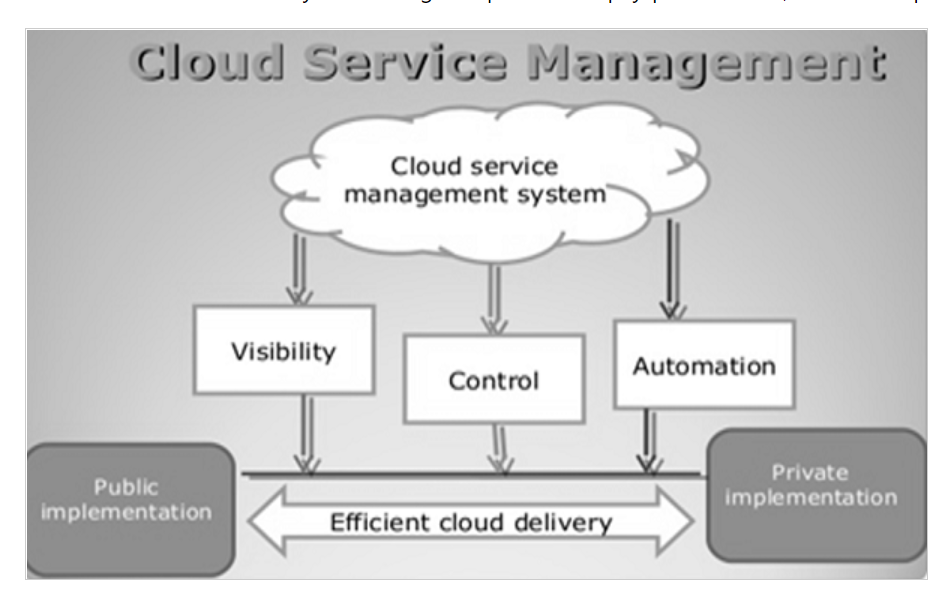
Clients of on-demand computing services essentially use these services as offsite virtual servers. Instead of investing in their own physical infrastructure, a company operates on a pay- as-we-go plan with a cloud services provider. On-demand computing itself is not a new concept, but has acquired new life thanks to cloud computing. In previous years, on-demand computing was provided from a single server via some sort of time-sharing arrangement. Today, the service is based on large grids of computers operating as a single cloud.

# 2.8 Cloud Service Management

The management of cloud infrastructure products and services is cloud management. Public clouds are operated by public cloud service providers, which provide the servers, storage, networking and data centre operations of the public cloud environment. With a third-party cloud management tool, users can also choose to manage their public cloud services.

Public cloud service users can typically choose from three categories of specific cloud provisioning:

* **User self-provisioning**: Users, usually via a web form or console interface, buy cloud services directly from the provider. On a per-transaction basis, the client pays.
* **Advanced provisioning**: A pre-determined sum of services scheduled in advance of operation is contracted in advance by customers. A flat fee or a monthly fee is charged by the consumer.
* **Dynamic provisioning**: When the client requires them, the provider allocates resources, and then decommissions them when they are no longer required. On a pay-per-use basis, the client is paid.



**Fig 2.22 Cloud Service Management**

The purpose and scope of the management of cloud services are listed below:

* **Purpose**: Establish suitable techniques for managing and running cloud-based services. Insert cloud service management techniques into current frameworks for IT creation and support.
* **Scope**: Oversight of cloud-based service design, development and change. Cloud-based service management and operation.

## Characteristics of Cloud service Management

In a design for handling cloud environments, cloud management incorporates applications and technologies. With a range of cloud management platforms and instruments, software developers have responded to the management challenges of cloud computing. These solutions include native tools provided by public cloud providers, as well as third-party tools designed by various cloud providers to provide consistent functionality. With access to various native features within individual cloud platforms, administrators must balance the conflicting requirements of efficient consistency across various cloud platforms. The need for transparent cross-platform management is motivated by increasing public cloud adoption and increased multi-cloud use. For those technical professionals responsible for maintaining IT systems and facilities, the rapid adoption of cloud services presents a new set of management challenges.

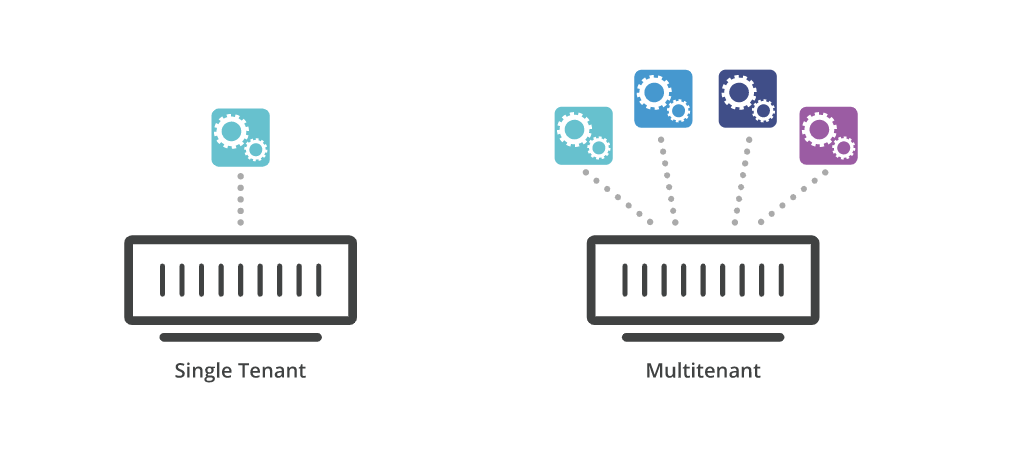
In the following categories, cloud-management systems and instruments should be able to have minimum functionality.

* **Service request**: receiving and fulfilling user requests to access and deploy cloud services.
* **Cost management and optimization**: Cloud spending monitors and accurate sizes and aligns resources and efficiency with real demand.
* **Security and compliance**: handling cloud providers' role-based access and implementing security settings.
* **Inventory and classification**: discover and maintain pre-existing cloud infrastructure in the brownfield plus track and handle modifications.

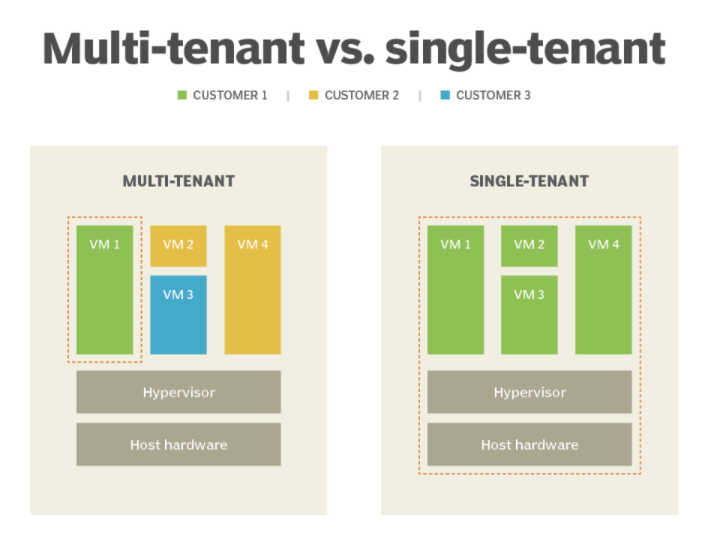
**2.9 Multitenancy**

In cloud computing, multitenancy means that multiple customers of a cloud vendor are using the same computing resources. Despite the fact that they share resources, cloud customers aren't aware of each other, and their data is kept totally separate. Multitenancy is a crucial component of cloud computing; without it, cloud services would be far less practical. Multitenant architecture is a feature in many types of public cloud computing, including IaaS, PaaS, SaaS, containers, and serverless computing.

**Single tenant and multitenant**



**Fig 2.23 a Single and multitenant architectures**



**Fig 2.23 b Single and multitenant architectures**

To understand multitenancy, think of how banking works. Multiple people can store their money in one bank, and their assets are completely separate even though they're stored in the same place. Customers of the bank don't interact with each other, don't have access to other customers' money, and aren't even aware of each other. Similarly, in public cloud computing, customers of the cloud vendor use the same infrastructure – the same servers, typically – while still keeping their data and their business logic separate and secure.

The classic definition of multitenancy was a single software instance\* that served multiple users, or tenants. However, in modern cloud computing, the term has taken on a broader meaning, referring to shared cloud infrastructure instead of just a shared software instance.

*\**A software instance is a copy of a running program loaded into random access memory (RAM).

**Multitenancy and cloud computing**

In cloud computing, applications and data are hosted in remote servers in various data centers and accessed over the Internet. Data and applications are centralized in the cloud instead of being located on individual client devices (like laptops or smartphones) or in servers within a company's offices.

Many modern applications are cloud-based, which is why, for example, a user can access their Facebook account and upload content from multiple devices.

**Benefits of Multitenancy**

Many of the benefits of cloud computing are only possible because of multitenancy. Here are two crucial ways multitenancy improves cloud computing:

**Better use of resources:** One machine reserved for one tenant isn't efficient, as that one tenant is not likely to use all of the machine's computing power. By sharing machines among multiple tenants, use of available resources is maximized.

**Lower costs:** With multiple customers sharing resources, a cloud vendor can offer their services to many customers at a much lower cost than if each customer required their own dedicated infrastructure.

**Drawbacks of Multitenancy**

**Possible security risks and compliance issues:** Some companies may not be able to store data within shared infrastructure, no matter how secure, due to regulatory requirements. Additionally, security problems or corrupted data from one tenant could spread to other tenants on the same machine, although this is extremely rare and shouldn't occur if the cloud vendor has configured their infrastructure correctly. These security risks are somewhat mitigated by the fact that cloud vendors typically are able to invest more in their security than individual businesses can.

**The "noisy neighbor" effect:** If one tenant is using an inordinate amount of computing power, this could slow down performance for the other tenants. Again, this should not occur if the cloud vendor has set up their infrastructure correctly.

**Case study on cloudfare solutions to support multitenancy in different categories**

1. **Mutlitenancy support in public cloud computing**

Imagine a special car engine that could be shared easily between multiple cars and car owners. Each car owner needs the engine to behave slightly differently: some car owners require a powerful 8-cylinder engine, while others require a more fuel-efficient 4-cylinder engine. Now imagine that this special engine is able to morph itself each time it starts up so that it can better meet the car owner's needs.

This is similar to the way many public cloud providers implement multitenancy. Most cloud providers define multitenancy as a shared software instance. They store metadata\* about each tenant and use this data to alter the software instance at runtime to fit each tenant's needs. The tenants are isolated from each other via permissions. Even though they all share the same software instance, they each use and experience the software differently.

1. **Multitenancy support in container architecture**

Containers are self-contained bundles of software that include an application, system libraries, system settings, and everything else the application needs in order to run. Containers help ensure that an application runs the same no matter where it is hosted.

Containers are partitioned from each other into different user space environments, and each container runs as if it were the only system on that host machine. Because containers are self-contained, multiple containers created by different cloud customers can run on a single host machine.

1. **Multitenancy support in serverless computing**

Serverless computing is a model in which applications are broken up into smaller pieces called functions, and each function only runs on demand, separately from the other functions. (This model of cloud computing is also known as Function-as-a-Service, or FaaS.)

As the name implies, serverless functions do not run on dedicated servers, but rather on any available machine in the serverless provider's infrastructure. Because companies are not assigned their own discrete physical servers, serverless providers will often be running code from several of their customers on a single server at any given time – another example of multitenancy.

Some serverless platforms use Node.js for executing serverless code. The Cloudflare serverless platform, Cloudflare Workers, uses Chrome V8, in which each function runs in its own sandbox, or separate environment. This keeps serverless functions totally separate from each other even when they’re running on the same infrastructure.

1. **Multitenancy support in private cloud computing**

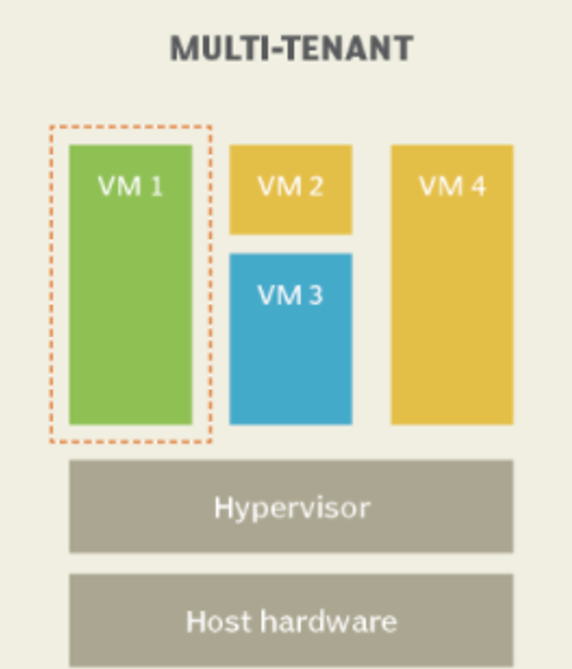
Private cloud computing uses multitenant architecture in much the same way that public cloud computing does. The difference is that the other tenants are not from external organizations. In public cloud computing, Company A shares infrastructure with Company B. In private cloud computing, different teams within Company A share infrastructure with each other.

**2.10 Multitenant cloud architecture**

A **multitenant cloud architecture** describes a single cloud instance and infrastructure purpose-built to support multiple customers.

**Multitenancy** can describe hardware or software architectures in which multiple systems, applications, or data from different enterprises are hosted on the same physical hardware. This differs from single-tenancy, in which a server runs one instance of an operating system and application. In the cloud world, a multitenant cloud architecture enables customers (“tenants”) to share computing resources in a public or private cloud.

Multitenancy is a common feature of purpose-built, cloud-delivered services, as it allows customers to share resources efficiently while securely scaling to meet increasing demand. Despite the fact that they share resources, cloud customers aren't aware of each other and their data is kept totally separate.



**Fig 2.4 Multitenant architecture**

## References

1. Dennis Gannon and Dan Reed , Parallelism and the Cloud, October 16, 2009,

<https://www.drdobbs.com/parallel/parallelism-and-the-cloud/220601206>

2 . Cloud Service management: https://www.includehelp.com/cloud-computing/cloud-service-management.aspx

1. Web Services: https://[www.cleo.com/blog/knowledge-base-web-services](http://www.cleo.com/blog/knowledge-base-web-services)
2. WSDL: https://[www.w3schools.com/xml/xml\_wsdl.asp](http://www.w3schools.com/xml/xml_wsdl.asp)
3. UDDI: https://[www.tutorialspoint.com/uddi/uddi\_overview.htm](http://www.tutorialspoint.com/uddi/uddi_overview.htm)

Cloud types: https://[www.vxchnge.com/blog/different-types-of-cloud-computing](http://www.vxchnge.com/blog/different-types-of-cloud-computing)