UNIT IV- CLOUD ENABLING TECHNOLOGIES

Service Oriented Architecture – Web Services – Basics of Virtualization – Emulation – Types of Virtualizations – Implementation levels of Virtualization – Virtualization structures – Tools & Mechanisms – Virtualization of CPU, Memory & I/O Devices – Desktop Virtualization – Server Virtualization – Google App Engine – Amazon AWS - Federation in the Cloud.

4.1. SERVICE ORIENTED ARCHITECTURE:

Service-Oriented Architecture (SOA) in cloud computing is a design paradigm that structures applications as a collection of loosely coupled, reusable, and independent services accessible over a network.

It promotes agility, scalability, and integration by using defined interfaces to enable communication between disparate systems. Key components include services, registries for discovery, and an enterprise service bus (ESB) for integration.

Core Concepts of SOA in Cloud

- Services: Discrete, self-contained units of functionality (e.g., payment processing, data validation).
- Loose Coupling: Services interact with minimal dependencies, allowing them to be updated or replaced without affecting other components.
- Interoperability:

SOA enables different applications, technologies, and platforms to communicate seamlessly

- Reusability: Services are designed to be used across multiple applications, reducing development effort.
- Service Registry: A catalog that helps developers discover and invoke services at runtime.

Benefits of Combining SOA with Cloud

- Increased Agility: Businesses can quickly adapt by reconfiguring services.
- Cost Efficiency: Reusable services eliminate the need to build functionality from scratch.
- Scalability: Services can be scaled independently in a cloud environment.
- Simplified Integration: Connects legacy systems with modern cloud services.

Challenges and Risks

- Overcomplexity: Creating too many small services can lead to an unmanageable system.
- Security & Reliability: Reliance on third-party cloud providers can lead to issues with data security and service availability.
- Integration Hurdles: Migrating existing, complex systems into a cloud-based SOA environment can be difficult.

SOA vs. Microservices

While both promote modularity, SOA is typically enterprise-wide, often using an Enterprise Service Bus (ESB) to connect large-scale, heterogeneous applications. Microservices are generally more granular, lightweight, and focused on specific, small-scale functionalities within a single application.

Definition:

Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration. It defines a way to make software components reusable using the interfaces.

Formally, SOA is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a network call over the internet.

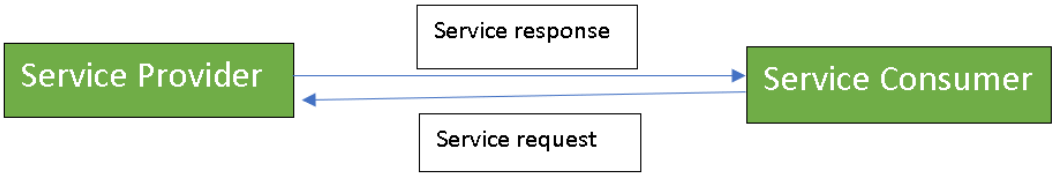
It uses common communication standards to speed up and streamline the service integrations in applications. Each service in SOA is a complete business function

in itself. The services are published in such a way that it makes it easy for the developers to assemble their apps using those services. Note that SOA is different from microservice architecture.

- SOA allows users to combine a large number of facilities from existing services to form applications.
- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

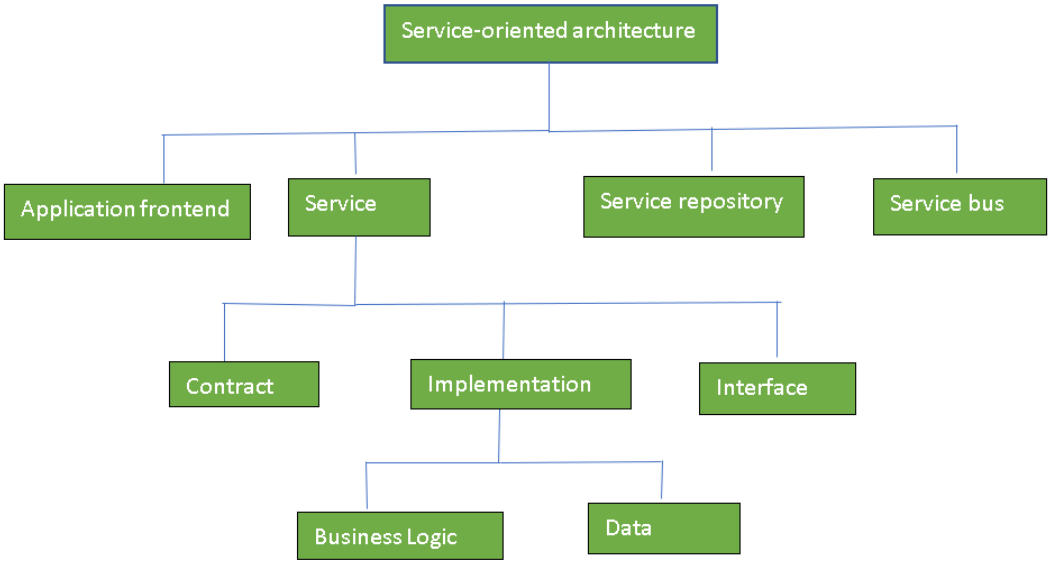
The different characteristics of SOA are as follows:
o Provides interoperability between the services.
o Provides methods for service encapsulation, service discovery, service composition,
service reusability and service integration.
o Facilitates QoS (Quality of Services) through service contract based on Service Level
Agreement (SLA).
o Provides loosely couples services.
o Provides location transparency with better scalability and availability.
o Ease of maintenance with reduced cost of application development and deployment.
There are two major roles within Service-oriented Architecture:

1. Service provider: The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.
2. Service consumer: The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.



Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer. This practice is known as service orchestration Another important interaction pattern is service choreography, which is the coordinated interaction of services without a single point of control.

Components of SOA:



Guiding Principles of SOA:

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

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

Advantages of SOA:

- Service reusability: In SOA, applications are made from existing services. Thus, services can be reused to make many applications.
- Easy maintenance: As services are independent of each other they can be updated and modified easily without affecting other services.
- Platform independent: SOA allows making a complex application by combining services picked from different sources, independent of the platform.
- Availability: SOA facilities are easily available to anyone on request.
- Reliability: SOA applications are more reliable because it is easy to debug small services rather than huge codes
- Scalability: Services can run on different servers within an environment, this increases scalability

Disadvantages of SOA:

- High overhead: A validation of input parameters of services is done whenever services interact this decreases performance as it increases load and response time.
- High investment: A huge initial investment is required for SOA.

- Complex service management: When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

Practical applications of SOA: SOA is used in many ways around us whether it is mentioned or not.

1. SOA infrastructure is used by many armies and air forces to deploy situational awareness systems.
2. SOA is used to improve healthcare delivery.
3. Nowadays many apps are games and they use inbuilt functions to run. For example, an app might need GPS so it uses the inbuilt GPS functions of the device. This is SOA in mobile solutions.
4. SOA helps maintain museums a virtualized storage pool for their information and content.

How does service-oriented architecture work?

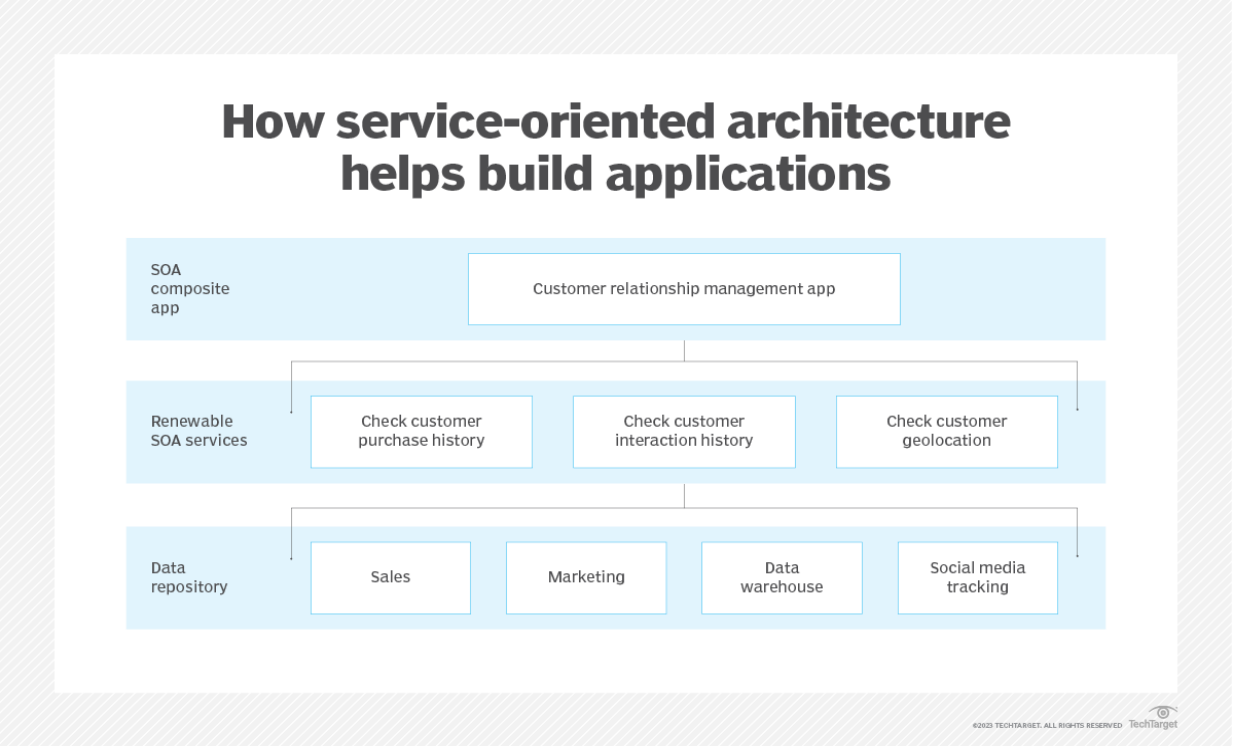
SOA simplifies complex software systems into reusable services that can be accessed by other applications and users referred to as service consumers. These services can be used as building blocks of new applications. Each SOA service has a specific task and an interface that includes the service's input and output parameters as well as the communication protocol required to access it.

SOA lets services communicate using a loose coupling system to either pass data or coordinate an activity. Loose coupling refers to a client of a service remaining independent of the service it requires. In addition, the client -- which can also be a service -- can communicate with other services, even when they aren't related. Through this process, various services can be combined to create more complex software that other applications can use as a single unit.

A consumer or application owner sends input data to request information or a task from a service. The service processes the data or performs its requested task and sends back a response. For example, an individual in sales or marketing could perform an SOA service request from a customer relationship management system, which provides access to customer data. The employee can

give the service the relevant input, such as a specific customer's name, and it will return the requested response, which might include the customer's purchase history.

SOA is an implementation of the service concept or service model of computing. In this architectural style, business functions and processes are implemented as software services, accessed through a set of strictly defined application programming interfaces (APIs) and bound into applications through dynamic service orchestration.



4.2. **WEB SERVICES**

The Internet is the worldwide connectivity of hundreds of thousands of computers of various types that belong to multiple networks. On the World Wide Web, a web service is a standardized method for propagating messages between client and server applications. A web service is a software module that is intended to carry out a specific set of functions.

Web services in cloud computing can be found and invoked over the network.The web service would be able to deliver functionality to the client that invoked the web service.

A web service is a set of open protocols and standards that allow data to be exchanged between different applications or systems.

Web services can be used by software programs written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the Internet in a similar way to inter-process communication on a single computer.

Any software, application, or cloud technology that uses standardized web protocols (HTTP or HTTPS) to connect, interoperate, and exchange data messages – commonly XML (Extensible Markup Language) – across the internet is considered a web service.
Web services have the advantage of allowing programs developed in different languages to connect with one another by exchanging data over a web service between clients and servers. A client invokes a web service by submitting an XML request, which the service responds with an XML response**.**

Functions of Web Services

- It's possible to access it via the internet or intranet networks.
- XML messaging protocol that is standardized.
- Operating system or programming language independent.
- Using the XML standard, it is self-describing.
- A simple location approach can be used to locate it.

Components of Web Service

XML and HTTP is the most fundamental web services platform. The following components are used by all typical web services:

SOAP (Simple Object Access Protocol)

SOAP stands for "Simple Object Access Protocol." It is a transport-independent messaging protocol. SOAP is built on sending XML data in the form of SOAP Messages. A document known as an XML document is attached to each message. Only the structure of the XML document, not the content, follows a pattern. The best thing about Web services and SOAP is that everything is sent through HTTP, the standard web protocol.

A root element known as the element is required in every SOAP document. In an XML document, the root element is the first element. The "envelope" is separated into two halves. The header comes first, followed by the body. The routing data,

or information that directs the XML document to which client it should be sent to, is contained in the header. The real message will be in the body.

UDDI (Universal Description, Discovery, and Integration)

UDDI is a standard for specifying, publishing and discovering a service provider's online services.

It provides a specification that aids in the hosting of data via web services.

UDDI provides a repository where WSDL files can be hosted so that a client application can discover a WSDL file to learn about the various actions that a web service offers.

As a result, the client application will have full access to the UDDI, which serves as a database for all WSDL files.
The UDDI registry will hold the required information for the online service, just like a telephone directory has the name, address, and phone number of a certain individual. So that a client application may figure out where it is.

WSDL (Web Services Description Language)

If a web service can't be found, it can't be used. The client invoking the web service should be aware of the location of the web service.

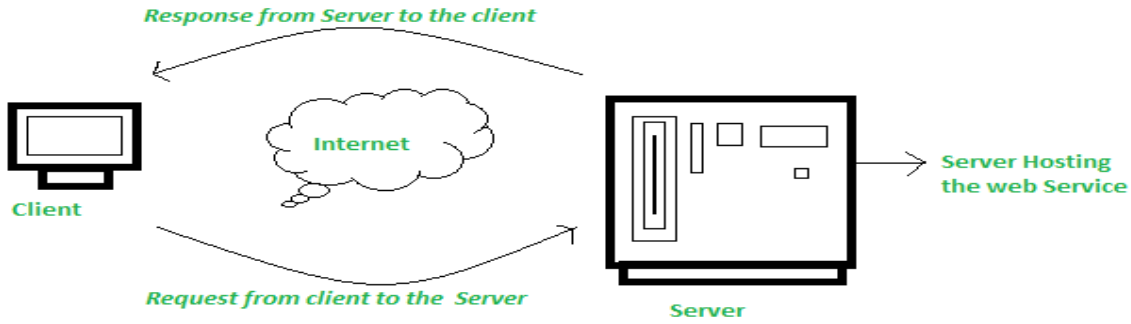
Second, the client application must understand what the web service does in order to invoke the correct web service.

The WSDL, or Web services description language, is used to accomplish this. The WSDL file is another XML-based file that explains what the web service does to the client application.

The client application will be able to understand where the web service is located and how to use it by using the WSDL document.

How Does Web Service Work?

The diagram depicts a very simplified version of how a web service would function. The client would use requests to send a sequence of web service calls to a server that would host the actual web service.



Remote procedure calls are what are used to make these requests. Calls to methods hosted by the relevant web service are known as Remote Procedure Calls (RPC). Example: Flipkart offers a web service that displays prices for items offered on Flipkart.com. The front end or presentation layer can be written in .Net or Java, but the web service can be communicated using either programming language.

The data that is exchanged between the client and the server, which is XML, is the most important part of a web service design. XML (Extensible markup language) is a simple intermediate language that is understood by various programming languages. It is a counterpart to HTML. As a result, when programs communicate with one another, they do so using XML.

This creates a common platform for applications written in different programming languages to communicate with one another. For transmitting XML data between applications, web services employ SOAP (Simple Object Access Protocol).

The data is sent using standard HTTP. A SOAP message is data that is sent from the web service to the application. An XML document is all that is contained in a SOAP message. The client application that calls the web service can be created in any programming language because the content is written in XML.

Features/Characteristics Of Web Service

Web services have the following features:

(a) XML Based: The information representation and record transportation layers of a web service employ XML. There is no need for networking, operating system, or platform binding when using XML. At the middle level, web offering-based applications are highly interoperable.

(b) Loosely Coupled: A customer of an internet service provider isn't necessarily directly linked to that service provider. The user interface for a web service provider can change over time without impacting the user's ability to interact with the service provider. A strongly coupled system means that the patron's and server's decisions are inextricably linked, indicating that if one interface changes, the other should be updated as well.
A loosely connected architecture makes software systems more manageable and allows for easier integration between different structures.

(c) Capability to be Synchronous or Asynchronous: Synchronicity refers to the client's connection to the function's execution. The client is blocked and the client has to wait for the service to complete its operation, before continuing in synchronous invocations. Asynchronous operations allow a client to invoke a task and then continue with other tasks.
Asynchronous clients get their results later, but synchronous clients get their effect immediately when the service is completed. The ability to enable loosely linked systems requires asynchronous capabilities.

(d) Coarse-Grained: Object-oriented systems, such as Java, make their services available through individual methods. At the corporate level, a character technique is far too fine an operation to be useful. Building a Java application from the ground, necessitates the development of several fine-grained strategies, which are then combined into a rough-grained provider that is consumed by either a buyer or a service.
Corporations should be coarse-grained, as should the interfaces they expose. Web services generation is an easy approach to define coarse-grained services that have access to enough commercial enterprise logic.

(e) Supports Remote Procedural Call: Consumers can use an XML-based protocol to call procedures, functions, and methods on remote objects utilizing web services. A web service must support the input and output framework exposed by remote systems.
Enterprise-wide component development Over the last few years, JavaBeans (EJBs) and.NET Components have become more prevalent in architectural and enterprise deployments. A number of RPC techniques are used to allocate and access both technologies.
A web function can support RPC by offering its own services, similar to those of a traditional role, or by translating incoming invocations into an EJB or.NET component invocation.

(f) Supports Document Exchanges: One of XML's most appealing features is its simple approach to communicating with data and complex entities. These

records can be as simple as talking to a current address or as complex as talking to an entire book or a Request for Quotation. Web administrations facilitate the simple exchange of archives, which aids incorporate reconciliation. The web benefit design can be seen in two ways: **(i)** The first step is to examine each web benefit on-screen character in detail. **(ii)** The second is to take a look at the rapidly growing web benefit convention stack.

Advantages Of Web Service

Using web services has the following advantages:

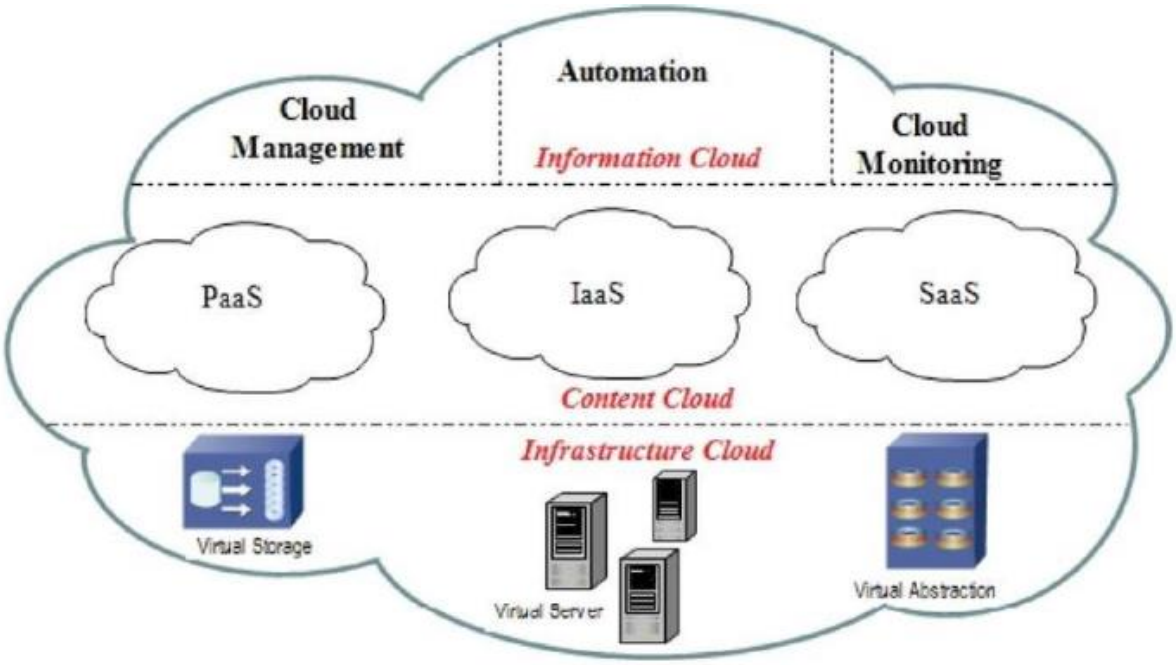
(a) Business Functions can be exposed over the Internet: A web service is a controlled code component that delivers functionality to client applications or end-users. This capability can be accessed over the HTTP protocol, which means it can be accessed from anywhere on the internet. Because all apps are now accessible via the internet, Web services have become increasingly valuable. Because all apps are now accessible via the internet, Web services have become increasingly valuable. That is to say, the web service can be located anywhere on the internet and provide the required functionality.

(b) Interoperability: Web administrations allow diverse apps to communicate with one another and exchange information and services. Different apps can also make use of web services. A .NET application, for example, can communicate with Java web administrations and vice versa. To make the application stage and innovation self-contained, web administrations are used.

(c) Communication with Low Cost: Because web services employ the SOAP over HTTP protocol, you can use your existing low-cost internet connection to implement them. Web services can be developed using additional dependable transport protocols, such as FTP, in addition to SOAP over HTTP.

(d) A Standard Protocol that Everyone Understands: Web services communicate via a defined industry protocol. In the web services protocol stack, all four layers (Service Transport, XML Messaging, Service Description, and Service Discovery) use well-defined protocols.

(e) Reusability: A single web service can be used simultaneously by several client applications.



General Architecture of Cloud Web Services

How Do Web Services Work?

1. Service Description (WSDL)

- **Publishing the Service:** According to the need of the service provider, the provider needs to publish the web service with WSDL (Web Services Description Language). This information contains the name of the service – or its location, the operations that enable it, the messages passed between such services, and the data types employed.
- **WSDL Document**: This document represents a legal and digital agreement between the specific service provider and the service consumer as to the ways of accessing the service.

2. Service Discovery (UDDI)

- **Registering the Service:** To use web service, the service provider first needs to list his service in the UDDI (Universal Description, Discovery, and Integration). This is like a registry that shows where Web services can be published and found.
- **Finding the Service:** UDDI is used to register and discover a web service where the service consumer has to search for the most appropriate match. The Ws registry contains information about the application services and their respective WSDL.

3. Service Invocation (SOAP/REST)

SOAP-based Web Services:

- **Request Creation:** The service consumer, to execute the service, will generate a SOAP request message. Communicated below is the content of the message in XML and it is compliant with the structure defined within the WSDL document.
- **Sending the Request:** The SOAP message is transferred over a network protocol over the web to the service provider who has an endpoint.
- **Processing the Request:** The service provider then takes the SOAP request and completes the required service on the message received from the requester.
- **Response Creation**: After it performs the requested operation, the service provider has to construct a SOAP response message to return it to the consumer.

RESTful Web Services:

- **HTTP Methods:** Remisable services let developers leverage standard HTTP operations such as GET, POST, PUT and DELETE. Each method represents the type of performed operation on the data – read, write, modify, delete.
- **Request Creation:** The service consumer forges an HTTP request usually with a JSON containing an expected language in the body of the request and specifying a URI for the target resource.
- **Sending the Request:** This call is made in the form of an HTTP request sent to the endpoint of the service provider.
- **Processing the Request:** The end-user's HTTP request reaches the service provider where the server processes the request and performs the necessary operation to respond.
- **Response Creation:** The service provider returns an HTTP response in most cases, which includes a JSON body that contains the requested data or the status of the process.

4. Data Interchange (XML/JSON)

- **XML:** This specification is mainly applied to SOAP-based web services; however, XML formats the request-response messages in such a way that they can be easily readable as well as programmatically processible.

- **JSON:** As far as its application is concerned, JSON is mainly utilized in RESTful web services but it is quite compact and is relatively easier to parse as compared to XML due to these reasons the JSON format of data is quite suitable for web or mobile applications.

5. Service Security

- **SSL/TLS:** Protect messages transferred between the consumer of a service and the provider by enciphering them.
- **WS-Security:** Safeguards the SOAP messages by enhancing the protocol through aspects such as authentication, encryption, and digital signatures.
- **OAuth:** Mostly implemented in RESTful services to address a secure paradigm to control access to the user information so that third-party applications can access them without involving credentials.
- **JWT:** JSON Web Token is a compact, URL-safe means of representing claims to be transferred among two parties in a portable data structure payload.

Features of Web Services

1. Interoperability

- **Cross-Platform Communication:** The interoperation of applications within the integrated application incurs applications that are developed for two or more platforms and are coded in different languages. Web services help in this process.
- **Standardized Protocols:** They employ standard protocols which include: HTTP, XML, SOAP, and WSDL which enhance compatibility between the realms.

2. Extensibility

- **Flexible Integration:** One major advantage of web services is that they are highly interoperable and portable hence they can easily be incorporated into new applications in an organization without much alteration to existing systems and procedures.
- **Modular Design:** They are supposed to be developed in a rather logical manner, meaning that new features may be incorporated into the system without influencing any of the services.

3. Scalability

- **Distributed Computing:** Web services are easily portable over different platforms and can be implemented on different server instances so that issues with load balancing and failovers can be easily addressed.
- **Horizontal Scaling:** They can be scaled horizontally towards bringing in more load by adding another instance of the service.

4. Reusability

- **Service Reusability:** Web services enable medium and large companies to modularize their business processes and use them in different applications with similar characteristics.
- **Component Reuse:** Services can be built on existing services making efficient reuse of software in the form of available services possible.

5. Loose Coupling

- **Minimal Dependency:** Web services are inherently asynchronous so that the Web service consumer and the Web service provider do not require entailment of implementations.
- **Independent Deployment:** This means that when a subtype is used instead of a more general type of service implementation on the server side, it is not necessary to change the client application as long as the interface is the same.

6. Discoverability

- **UDDI Registries:** To look for a particular service the services available can be registered and searched through UDDI registries, for the available services.
- **Dynamic Binding:** The consumers are also able to procure the services dynamically and offer their requests at runtime.

7. Security

- **SSL/TLS:** It is necessary to note that web services can employ SSL/TLS to provide secure communication over the World Wide Web.
- **WS-Security:** With the current services, we can achieve WS-Security standards such as message integrity, confidentiality and authentication with the use of SOAP-based services.
- **OAuth:** OAuth can be used for secure authorization in RESTful services.

- **JWT:** This is utilized in trustworthy messaging concerning RESTful services and identification.

8. Standardized Messaging

- **SOAP:** SOAP now has a messaging protocol that is articulated in XML which helps in preserving the structure and manner in which messages are handled.
- **REST:** REST employs conventional HTTP methods and status assertion, which contributes towards the reduction of the interaction model.

9. Support for Complex Operations

- **Transaction Management:** This means that web services can support multi-step and other types of complex transactions, which will guarantee the right flow of processes.
- **Asynchronous Processing:** They can operate asynchronously, and this can be useful when dealing with long-running operations since they can be worked on in the background.

10. Versatility

- **Synchronous and Asynchronous:** Web services may be either the request/response model, where a Web service sends back confirmation of the receipt of a request and information regarding how the request can be completed or the message-oriented model, where the two elements exchange messages independent of each other.
- **Various Payload Formats:** It works with different payloads, though most applications use either XML or JSON ones depending on the peculiarities of the given program.

11. Platform Independence

- **Language Agnostic:** The creation and use of Web services does not require the developer to use any specific language but should support Web standards.
- **Protocol Agnostic:** Web services can run on different transports although, the most typical transport is HTTP; other transports include SMTP and JMS.