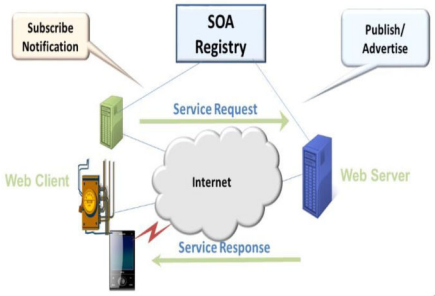
BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Service Oriented Architecture - Overview

- A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties
  - Logical view
  - Message Orientation
  - Description Orientation
  - Granularity
  - Network Orientation
  - Platform Neutrality
- Implementation of SOA
  - Arbitrary Web services (SOAP/XML)
  - RESTful Web services
  - Apache Thrift

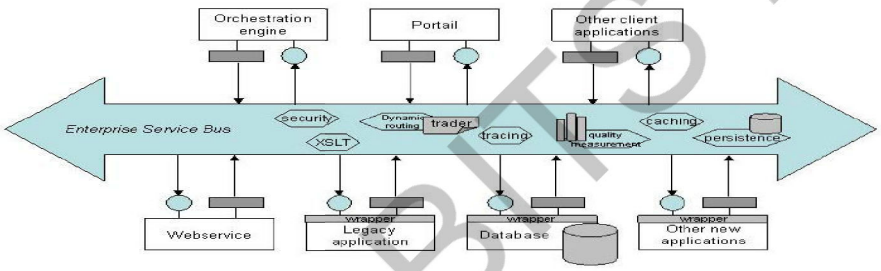


Characteristics of SOA

- Use of shared services — do not need to "reinvent the wheel"
- Loose coupling — can update applications with minimal effect on services that invoke them
- Location transparency — can re-host applications with minimal effect on services that invoke them
- Based on open standards — decreased dependence on vendor-specific solutions
- **Benefits of SOA**
- The services model mimics business needs better than objects or resources
  - IT Services (virus detection and elimination service)
  - Marketing services (promotional campaigns, brand marketing programs etc.)
  - Accounting services (Audits, Quarterly statements etc.)
- Loose coupling between consumer and provider
  - Allows us to improve availability by having redundant providers.
- Dynamic binding allows the use of the latest services that are most useful to the task at hand.
  - In a highly adhoc networked environment, "discovery" becomes critical – yellow pages model over white pages.
- Allows you to leverage existing IT assets – is a evolutionary architecture not a revolutionary one.

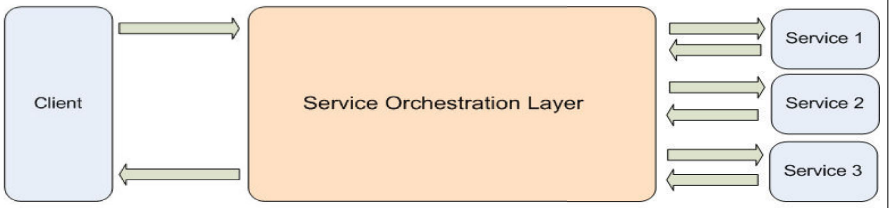
SOA vs ESB

- ESB - Implements a communication system between mutually interacting software applications (including SOA components)
- Can be used to integrate SOA apps with non-SOA apps
- Acts as service orchestrator, API gateway, security manager etc.



SOA Orchestration

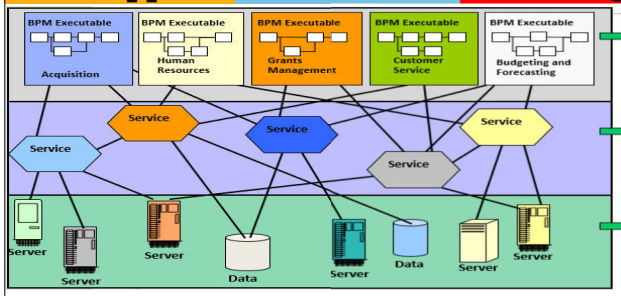
- Process of integrating two or more applications and/or services together to automate a process, or synchronize data in real-time
- Service orchestration is the combination of service interactions to create higher-level business services
- Service Orchestration involves connecting various applications, web services and legacy components in a business flow
- Examples of Orchestrators: Business Process Modeling engines, ESB etc.



innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

SOA supports Business Process Management



BPM executables implementing business processes access various information services to perform activities and manage workflow

A robust SOA provides the framework to define reusable services to support a wide range of business processes

Open standards enable leveraging of information assets from many agencies by "hiding" complexities of underlying agency infrastructures

The result: A high degree of flexibility and agility for government operations

CORBA has many complexities that are mitigated by SOA

Factor	*CORBA*	*SOA*
"Weight" of implementation	Heavyweight	More lightweight
Degree of coupling	Tight coupling (to components)	Loose coupling (between services and their underlying applications)
Communication Mode	Synchronous only	Synchronous or asynchronous
Initial investment	Large	Small-medium (depending on requirements)
Protocol type	Binary	Text
Processing "grain"	Fine-grained processing	Coarse-grained or fine-grained processing (depending on requirements)
Proprietary Level	Proprietary implementations	Non-proprietary implementations

It is important to note that services within an SOA can have CORBA components "behind" them

SOA Maturity Levels

Maturity level	SOA view	Benefits and metrics	Business involvement	Methodology	Service sourcing	Governance
Initial	Fine-grained software components	Promise of reuse; no metrics	Hidden from business	Minor adaptation of current software methods	Largely derived from existing application programming interfaces	Basic service definition policies at project level
Managed	Emerged software architecture	Standardization of data and resources	Services exposed as part of project cost	Project-level service definition methods	In-house development of new fine-grained software components	Service policies managed by registry monitors
Defined	Business process support	Business process redesign	Services part of requirement capture	Business service definition methods	External sourcing possible by defined function	Full set of service policies and metrics in place
Quantitatively managed	Enterprise service architecture	Agility and flexibility	Organizational "service thinking"	Intra- and inter-organizational service definition	Enterprise service bus to coordinate services	Business and IT governance metrics aligned
Optimized	Adaptive architecture	Autonomic systems	Value-stream "service thinking"	Value-chain service optimization	Full integration up and down the stack	Policy feedback used to adjust delivery

Thank You

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

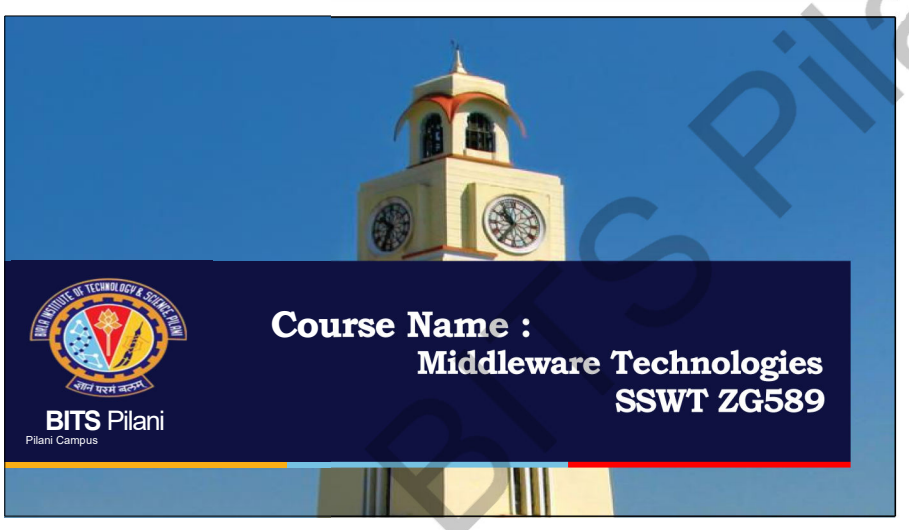
innovate achieve lead

Start Recording

innovate achieve lead

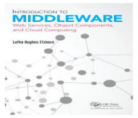
IMP Note to Students

- It is important to know that just login to the session does not guarantee the attendance.
- Once you join the session, continue till the end to consider you as present in the class.
- IMPORTANTLY, you need to make the class more interactive by responding to Professors queries in the session.
- **Whenever Professor calls your number / name ,you need to respond, otherwise it will be considered as ABSENT**



innovate achieve lead

Textbooks



T1: Letha Hughes Etzkorn - Introduction to middleware _ web services, object components, and cloud computing- Chapman and Hall_CRC (2017).

T2: William Grosso - Java RMI (Designing & Building Distributed Applications)

R1: Gregor Hohpe, Bobby Woolf - Enterprise Integration Patterns_ Designing, Building, and Deploying Messaging Solutions -Addison-Wesley Professional (2003)

R2: MongoDB in Action

Note: In order to broaden understanding of concepts as applied to Indian IT industry, students are advised to refer books of their choice and case-studies in their own organizations

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Modular Structure

No	Title of the Module
M1	Introduction and Evolution
M2	Enterprise Middleware
M3	Middleware Design and Patterns
M4	Middleware for Web-based Application and Cloud-based Applications
M5	Specialized Middleware

CS 10: Non-Rest Web Services

Agenda

- Web Services Types
- SOAP Messaging Protocol
- Web Services Description Language (WSDL)
- Java API for XML Web Services (JAX-WS)

Need for Web Services

- Modern day business applications use variety of programming platforms to develop web-based applications.
- Some applications may be developed in Java, others in .Net, while some other in Angular JS, Node.js, etc.
- Most often than not, these heterogeneous applications need some sort of communication to happen between them.
- Since they are built using different development languages, it becomes really difficult to ensure accurate communication between applications.
- Here is where web services come in.
- Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other.

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

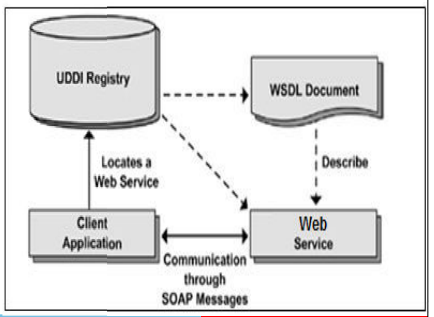
Web Services Types

- Two types of web services as follows:
- Non-REST (SOAP) Web Services (CS 10)
- RESTful Web Services (CS 11)

- Non-REST Web Services
  - First version of web services – released in late '90s
  - Implemented as arbitrary web services grouped together based on business context
  - Analogous to functional services – focus on functionality
  - Use SOAP (Simple Object Access Protocol) for data transfer over HTTP/S
  - Has embedded security via WS-Security header
  - Supports XML (SOAP is a restricted version of XML)

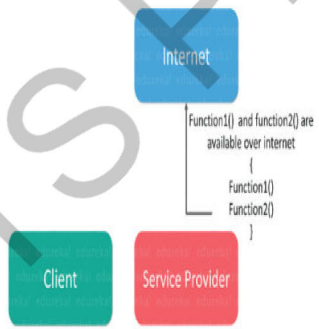
SOAP

- SOAP is an XML-based protocol to let applications exchange information over HTTP. Or more simple: SOAP is a protocol for accessing a Web Service.
- SOAP is a communication protocol
- SOAP is a format for sending messages
- SOAP is designed to communicate via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to get around firewalls
- SOAP is a W3C standard



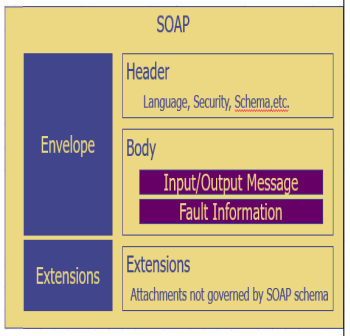
SOAP Services

- In the figure, you can notice how the client will communicate with the service provider. So in order to communicate, the client must know information like, the Location of web services server, Functions available, signature and return types of function and input-output formats.
- The service provider will create a standard XML file which will have all the above information. So, if this file is given to the client then it will be able to access web service.



SOAP Message Protocol

- Transfers structured information in the form of XML documents
- SOAP message is bound to the underlying protocol (mostly HTTP)
  - SOAP 1.1 – HTTP POST only
  - SOAP 1.2 (via web methods) – HTTP GET/POST/DELETE
- SOAP message goes from sender to receiver via intermediary nodes (SOAP nodes)
  - SOAP nodes may process SOAP headers and body or may add some additional headers for QoS / Security
- SOAP Message exchange patterns
  - Request-Reply
  - Response-only



SOAP Message

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

SOAP Message

- SOAP message is enclosed in SOAP envelope
- Header is optional, Body is mandatory
- Envelope will specify namespace used
  - SOAP 1.1 – "xmlns:soap=http://schemas.xmlsoap.org/soap/envelope"
  - SOAP 1.2 – "xmlns:soap=http://www.w3.org/2003/05/soap-envelope"
- Encoding
  - SOAP Encoding – uses SOAP defined XML schema
  - Literal Encoding – uses XML schema (recommended)
- Message Styles
  - RPC – SOAP Encoded
  - RPC – Literal
  - Document - Literal
  - Document – Literal Wrapped

```
<soap: Envelope>
        <soap:Header>
        …header stuff included here…
        </soap:Header>
        <soap:Body>
        …body stuff included here…
        </soap:Body>
</soap: Envelope>
```

```
<i xsi:type="xsd:int">7</i>
<1>/</1>
<hw: iElement>7</hw: iElement>
<hw:i >7</hw:i>
```


SOAP Message Structure

- The SOAP envelope element: Is the root element of a SOAP message. It contains the header and the <SOAP:BODY>element of the SOAP message.
- The SOAP header element: Is an optional part that gives extra information to the server and is the first descendent element of the SOAP envelope element.
- The SOAP body element: Is the compulsory part that contains the data of the message.
- The SOAP fault element: Is used to ascertain if there is a fault in the SOAP message and displays errors. This element describes the following four elements:
  - <faultcode>
  - <faultstring>
  - <faultactor>
  - <detail>

Skeleton SOAP Message in XML

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap=" ... ">
  <soap:Header>
      ... ...
  </soap:Header>
  <soap:Body>
      ... ...
      < … message payload goes here … >

  </soap:Body>
</soap:Envelope>
```


SOAP Header

- SOAP Header – Attributes
  - mustUnderstand
  - Role
  - Relay
- SOAP Fault – Contains error messages generated in processing

```
<soap:Header>
    <myschema:CurrentTransaction xmlns:myschema="http://myVeryOwnstuff/Transaction/"
        soap:role=predefined role URI (see Table 10.3) or custom, user defined Role
        soap:mustUnderstand="true"> … insert current data for the current transaction here…
    </myschema:CurrentTransaction>
</soap:Header>
```

```
<soap: Fault>
        <soap: Code>
                <soap:Value>
                        … put the appropriate code here. See Table 10.3 for codes
                </soap:Value>
                <soap:Subcode>
                        <soap:Value>
                        …your application specific subcode. Could be something like "Timeout"
                                or "InvalidRequest"
                        </soap:Value>
                </soap:Subcode>
        </soap:Code>
        <soap:Reason>
                <soap:Text xml:lang='en'>… a text sentence (in this case in English) saying
                more about the problem that occurred
                </soap:Text>
        </soap:Reason>
</soap: Fault>
```

Predefined Codes

- Version Mismatch
- MustUnderstand
- DataEncodingUnknown
- Sender
- Receiver
- Subcode

SOAP Request

```
<?xml version='1.0' encoding='utf-8'?>
<ENV:Envelope
  xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:m1="http://IONA.com/HelloWorld">
  <ENV:Body>
    <m1:greetMe >
      <stringParam0 xsi:type="xsd:string">
        Hello From WS Client
      </stringParam0>
    </m1:greetMe>
  </ENV:Body>
</ENV:Envelope>
```

SOAP Response

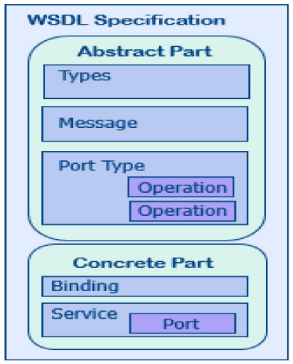
```
<?xml version='1.0' encoding='utf-8'?>
<ENV:Envelope
  xmlns:ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:m1="http://IONA.com/HelloWorld">
  <ENV:Body>
    <m1:greetMeResponse>
      <return xsi:type="xsd:string">
        Echo: Hello From WS Client
      </return>
    </m1:greetMeResponse>
  </ENV:Body>
</ENV:Envelope>
```


innovate achieve lead

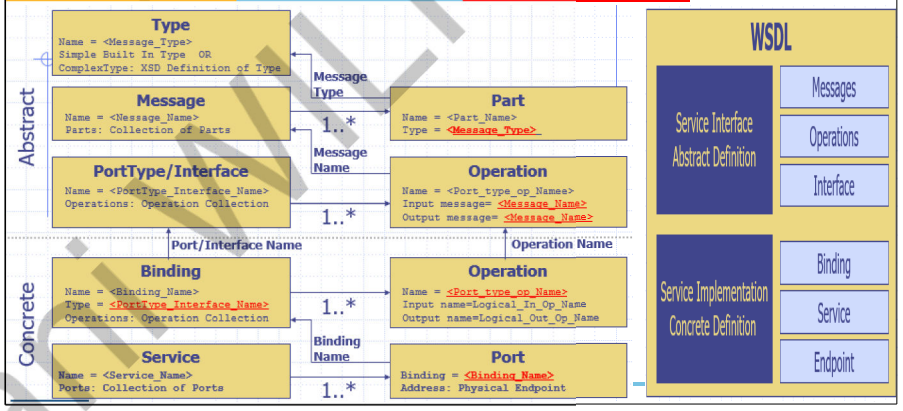
BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Web Services Description Language

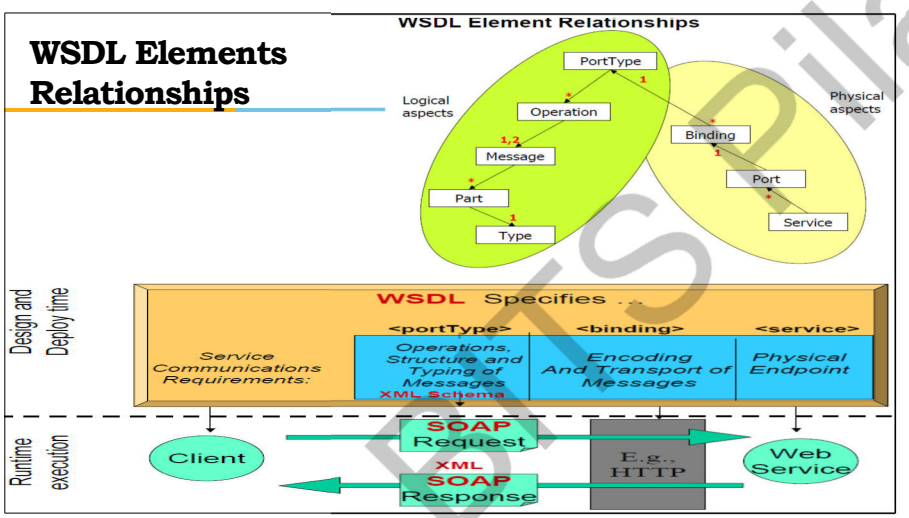
- WSDL stands for Web Services Description Language
- WSDL is based on XML
- WSDL is used to describe Web services
- WSDL is used to locate Web services
- WSDL is a W3C standard
- Defines the interface to a web service
  - Types
    - XML schema describing used data types
  - Messages
    - Necessary to invoke an operation of the service
  - Operations
    - Reference to input/output message
  - Port type
    - Set of operations that conform an instance of a service
  - Binding
    - Actual protocol to be used to invoke the operations
  - Services and ports
    - References to actual location of service



WSDL Specs



WSDL Elements Relationships



WSDL Specs Example

- **XML grammar**
  - <definitions>: root WSDL element
  - <types>: data types transmitted (starts with XML Schema specifications)
  - <message>: messages transmitted
  - <portType>: functions supported
  - <binding>: specifics of transmissions
  - <service>: how to access it

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

WSDL Specs Example – Walk through

```
<?xml version="1.0" ?>
-  <definitions name="TemperatureService"
   targetNamespace="http://www.xmethods.net/sd/TemperatureService.wsdl"
   xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  xmlns="http://schemas.xmlsoap.org/wsdl/">
    -  <message name="getTempRequest">
<part name="zipcode" type="xsd:string" />
</message>
    -  <message name="getTempResponse">
<part name="return" type="xsd:float" />
</message>
    -  <portType name="TemperaturePortType">
- <operation name="getTemp">
<input message="tns:getTempRequest" />
<output message="tns:getTempResponse" />
</operation>
</portType>
```

```
-<binding name="TemperatureBinding"
type="tns:TemperaturePortType">
<soap:binding style="rpc"
transport=http://schemas.xmlsoap.org/soap/http />
    -<operation name="getTemp">
<soap:operation soapAction="" />
        -<input>
<soap:body use="encoded" namespace="urn:xmethods-Temperature-Demo"  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</input>
-<output>
<soap:body use="encoded" namespace="urn:xmethods-Temperature-Demo"  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</output>
</operation>
</binding>
-<service name="TemperatureService">
<documentation>Returns current temperature in a given U.S. zipcode</documentation>
    -<port name="TemperaturePort"
    binding="tns:TemperatureBinding">
<soap:address  rpcrouter" />
</port>
</service>
</definitions>
```


JAVA API FOR XML WEB SERVICES (JAX-WS)

- Java API for creating SOAP web services, part of Java EE
- Server side code
- Service publisher
- Creating client with wsimport
  - **wsimport -keep http://localhost:8080/hello?wsdl -d .**

```
package myHelloWorld;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{

    @WebMethod String HelloWorld(String name);
}
```

```
package myHelloWorld;
import javax.jws.WebService;

//Service Implementation
@WebService(endpointInterface = "myHelloWorld.HelloWorld")
public class HelloWorldImpl implements HelloWorld{

    @Override
    public String HelloWorld(String name) {
        System.out.println(name+" says hello");
        return "Hello World " + name;
    }
}
```

```
package myHelloWorld;
import javax.xml.ws.Endpoint;
import myHelloWorld.HelloWorldImpl;

//Endpoint publisher
public class HelloWorldPublisher{

    public static void main(String[] args) {
        Endpoint.publish("http://localhost:8080/hello", new HelloWorldImpl());
    }
}
```

JAVA API FOR XML WEB SERVICES (JAX-WS)

- Client Stub

```
import myhelloworld.HelloWorld;
import myhelloworld.HelloWorldImplService;
public class HelloWorldClient {
    public static void main(String[] args) {
        HelloWorldImplService myHelloWorld = new HelloWorldImplService();
        HelloWorld myinterface = myHelloWorld.getHelloWorldImplPort();
        //Note the format of the operation call "helloWorld".
        //This matches the format in the wsimport-generated HelloWorld.java file.
        String response = myinterface.helloWorld(args[0]);

    System.out.println(response);
    }
}
```

- Dynamic Proxy Client

```
package myHelloWorld;

import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import myHelloWorld.HelloWorld;
/* This example does not use wsimport. Instead, it
creates a service instance manually. */

public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
        URL location_of_wsdl = new URL("http://localhost:8080/hello?wsdl");
        QName name_of_service = new QName("http://myHelloWorld/", "HelloWorldImplService");
        Service service = Service.create(location_of_wsdl, name_of_service);
        HelloWorld hello = service.getPort(HelloWorld.class);
        String response = hello.HelloWorld(args[0]);
        System.out.println(response);
    }
}
```

UDDI

- UDDI is a directory service where companies can register and search for Web services.
- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- A specification that defines
  - how to interact with a registry
  - What the entries on the registry look like
- Interaction with UDDI
  - Registration
    - Adding new service descriptions to the registry
  - Lookup
    - Queries to search for right services
- Types of UDDI registry
  - Public
    - Open search-engines for web services
  - Private
    - Created by companies for their own use

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

XML – RPC

- This is the simplest XML based protocol for exchanging information between computers.
- XML-RPC is a simple protocol that uses XML messages to perform RPCs.
- Requests are encoded in XML and sent via HTTP POST.
- XML responses are embedded in the body of the HTTP response.
- XML-RPC is platform-independent.
- XML-RPC allows diverse applications to communicate.
- XML-RPC is the easiest way to get started with web services.
- **Security:**
- Confidentiality

  If a client sends an XML request to a server, then question is that can we ensure that the communication remains confidential?
- Answer lies here
  - XML-RPC and SOAP run primarily on top of HTTP.
  - HTTP has support for Secure Sockets Layer (SSL).
  - Communication can be encrypted via the SSL.
  - SSL is a proven technology and widely deployed.

Security

- Authentication
- If a client connects to a web service, how do we identify the user? And is the user authorized to use the service?

- Following options can be considered but there is no clear consensus on a strong authentication scheme.
  - HTTP includes built-in support for Basic and Digest authentication, and services can therefore be protected in much the same manner as HTML documents are currently protected.
  - SOAP Security Extensions: Digital Signature (SOAP-DSIG). DSIG leverages public key cryptography to digitally sign SOAP messages. This enables the client or server to validate the identity of the other party.
  - The Organization for the Advancement of Structured Information Standards (OASIS) is working on the Security Assertion Markup Language (SAML, It is an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider).

Security

- Network Security
  - There is currently no easy answer to this problem, and it has been the subject of much debate. For now, if you are truly intent on filtering out SOAP or XML- RPC messages, one possibility is to filter out all HTTP POST requests that set their content type to text/xml.

- **Web Services Interoperability Organization (WS-I)**
- Web Services depend heavily on agreed-upon standards (XML, SOAP, WSDL, etc.) that are relatively independent of each other
- Keeping up with the changes and nuances that exist among these is difficult and time-consuming
- WS-I has the mission of doing this for the IT industry
  - advising on how to use these WS standards appropriately and interoperable
- Primary deliverables:
  - Basic Profile 1.0: this supplies advice about which combinations of the standards should be used together, and which features should be avoided or emphasized from each standard
  - Test Tools for verifying compliance with the Basic Profile
  - Sample Applications for assisting others in building compliant systems
  - Visit http://ws-I.org/ for more details

Non-REST vs RESTful Web services

Non-REST Web services (SOAP WS)	RESTful Web services
Exposes services as individual operations, each to achieve a specific business objective.	Representational State Transfer is an architectural style for defining interoperability in terms of resources or concepts
Introduced in W3C draft for web services in 2002	Introduced in W3C extended revision in 2004
Uses Simple Object Oriented Protocol (SOAP), a variant of XML for data transfer	REST supports SOAP/XML/JSON or any other data protocol supported by HTTP
Service contract definition is specified via Web service description language (WSDL)	Service contract is defined via resource or concept endpoints and HTTP methods (POST/ GET/ PUT/ DELETE) defined on them
JAX-WS is the binding protocol used	**JAX-RS** is binding protocol used
Imposes a rigid structure for service definition and request/response	Does not impose any specific structure – any format supported by HTTP is accessible
Requires more footprint and bandwidth, resulting in performance issues at large scale	Light-weight formats (like JSON) require less footprint there by improving performance
Defines its own security, as defined by SOAP-WS security	I**nherits security measures** from the underlying transport protocol (TCP, UDP etc.)

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

innovate achieve lead

Thank You

innovate achieve lead

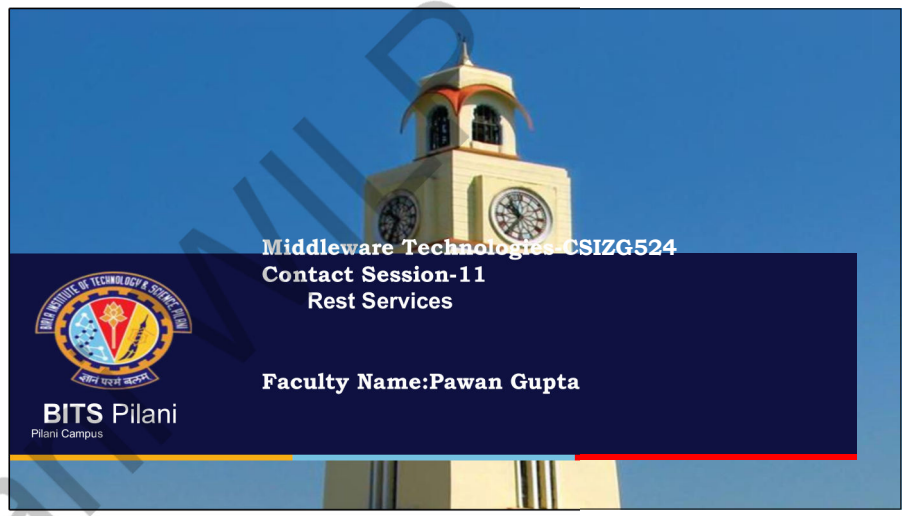
innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

IMP Note to Self

innovate achieve lead

Start Recording



Middleware Technologies CSIZG524
Contact Session-11
Rest Services

Faculty Name:Pawan Gupta

BITS Pilani
Pilani Campus

Important Note to Students

innovate achieve lead

- It is important to know that just login to the session does not guarantee the attendance.
- Once you join the session, continue till the end to consider your attendance.
- IMPORTANTLY, you need to make the class more interactive by responding to Professors queries in the session.
- Answering to Polls / Quiz questions during/after the class is mandatory to get attendance and the score will count for IA marks.
- **Whenever Professor calls your number / name, you need to respond, otherwise it will be considered as ABSENT**

innovate achieve lead

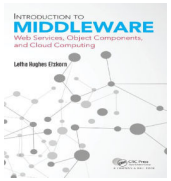
Disclaimer

- *The slides presented here are obtained from the authors of the books and from various other contributors.*
- *I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Text Books



T1: Letha Hughes Etzkorn - Introduction to middleware _ web services, object components, and cloud computing- Chapman and Hall_CRC (2017).

T2: William Grosso - Java RMI (Designing & Building Distributed Applications)

R1: Gregor Hohpe, Bobby Woolf - Enterprise Integration Patterns_ Designing, Building, and Deploying Messaging Solutions -Addison-Wesley Professional (2003)

R2: MongoDB in Action

Note: In order to broaden understanding of concepts as applied to Indian IT industry, students are advised to refer books of their choice and case-studies in their own organizations

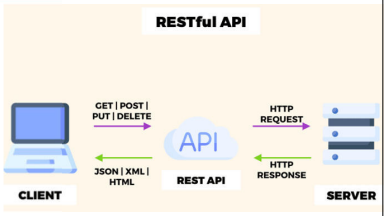
CS 11: Rest Services, JAX-RS Basics, Annotations and implementation

Agenda

- Rest Services
- JAX-RS Basics and Annotations
- JAX-RS implementations

RESTful Web Services

- REST – Architectural Constraints
- JAVA API FOR RESTFUL WEB SERVICES (JAX-RS)
  - Annotations
  - Server deployment
  - Client API
  - JSON API
- What Is a REST API?
  - An API (application programming interface) is a set of functions and protocols that enables communications between different software applications or systems.
  - The API establishes a collection of commands that users can invoke to retrieve the information that they're looking for in an organized, programmatic manner.



innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

RESTful API

- REST (representational state transfer) is a style of software architecture that defines constraints on how different systems communicate.
- An API that complies with these REST principles is known as a "REST API."
- What Is a REST API (RESTful API)?
- REST suggests to create an object of the data requested by the client and send the values of the object in response to the user.
- For example, if the user is requesting for a movie in City at a certain place and time, then you can create an object on the server side.
- REST stands for Representational State Transfer.
- It is a software architecture style that relies on a stateless communications protocol, most commonly, HTTP.
- REST structures data in XML, YAML, or any other format that is machine-readable, but usually JSON is most widely used.

RESTful Web Services

- Application programming interfaces (APIs) provide the platform and medium for applications to talk to and understand each other.
- APIs specify the way information passed across platforms is structured so that applications can exchange data and information. API is like an open language, the rules of which are shared by a certain service.
- 4 Commonly Used API Methods:
- GET: It provides read-only access to a resource.
- POST: It is used to create or update a new resource.
- PUT: It is used to update or replace an existing resource or create a new resource.
- DELETE: It is used to remove a resource

RESTful Web Services - Types

- There are four main types of APIs:
- **Open APIs:** Also known as Public APIs, there are no restrictions to access these types of APIs because they are publicly available.
- **Internal APIs:** Also known as Private APIs, only internal systems expose this type of API, which is, therefore, less known and often meant to be used inside the company. The company uses this type of API among the different internal teams to be able to improve its products and services.
- **Partner APIs:** One needs specific rights or licenses in order to access this type of APIs because they are not available to the public.
- **Composite APIs:** This type of API combines different data and service APIs.
- It is a sequence of tasks that run synchronously as a result of the execution and not at the request of a task.
- Its main uses are to speed up the process of execution and improve the performance of the listeners in the web interfaces.
- Examples of REST APIs:
  - Instagram API permits your applications to retrieve user tags, photos, account and much more.
  - Twitter also provides a REST API which a developer can query to source the latest tweets, or provide a search query that will return the results in JSON format.
  - GitHub also offers super REST API that you can utilize to perform actions such as following GitHub issues, tracking user activity, and create repositories from your app.

RESTful Web Services - principles

- The six principles of REST (five required, and one optional) are:
- **Client-server architecture:** REST APIs separate the client (the one requesting information) from the server (the one possessing information). The client does not have to worry about how the server stores and retrieves data.
- **Uniform interface:** Whether the client is a software application, a browser, a mobile app, or something else entirely, it can access and use the REST API in the same way.
- **Statelessness:** The server does not have to remember the client's state. All the client's requests must be "stateless," so each request must include all necessary information (such as the client's authentication details).
- **Cacheability:** REST servers can cache data and reuse it for other requests in the future.
- **Layered system:** REST APIs may have multiple intermediary layers between the client and the server. However, the client does not have to know these implementation details.
- **Code on demand (optional):** Clients may download code (e.g. Java applets or JavaScript scripts) to access additional functionality at runtime.

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

RESTful Web Services - principles

- Key principles around the HTTP and URI standards, sticking to which will make your HTTP application a RESTful-service-enabled application:
- Everything is a resource
- Each resource is identifiable by a unique identifier (URI)
- Use the standard HTTP methods
- Resources can have multiple representations
- Communicate statelessly

- **Principle 1 – Everything is a Resource**
- To understand this principle, one must conceive the idea of representing data by a specific format and not by a physical file.
- Each piece of data available on the Internet has a format that could be described by a content type.
- For example, JPEG Images; MPEG videos; html, xml, and text documents; and binary data are all resources with the following content types:
  - image/jpeg, video/mpeg, text/html,
  - text/xml, and application/octet-stream

RESTful Web Services - principles

- **Principle 2 – Each resource is identifiable by a unique identifier**
- Since the Internet contains so many different resources, they all should be accessible via URIs and should be identified uniquely. Furthermore, the URIs can be in a human readable format
- The URI keeps the data self-descriptive and eases further development on it.
- Here are a few sample examples of such URIs:
  - http://www.mydatastore.com/images/vacation/2014/summer
  - http://www.mydatastore.com/data/documents/balance?format=xml
  - http://www.mydatastore.com/data/archives/2014
- These human-readable URIs expose different types of resources in a straightforward manner.
- In the example, it is quite clear that the resource types are as follows:
  - Images, Videos, XML documents

- **Principle 3 – Use the Standard HTTP methods**
- The native HTTP protocol (RFC 2616) defines eight actions, also known as verbs:
  - GET, POST, PUT, DELETE
  - HEAD, OPTIONS, TRACE, CONNECT

RESTful Web Services - principles

- **Principle 3 – Use the Standard HTTP methods**
  - The first four of them feel just natural in the context of resources, especially when defining actions for resource data manipulation.
  - If you apply the REST principles correctly, the HTTP verbs should be used as shown here:

HTTP verb	Action	Response status code
GET	Request an existing resource	"**200 OK**" if the resource exists, "**404 Not Found**" if it does not exist, and "**500 Internal Server Error**" for other errors
PUT	Create or update a resource	"**201 CREATED**" if a new resource is created, "**200 OK**" if updated, and "**500 Internal Server Error**" for other errors
POST	Update an existing resource	"**200 OK**" if the resource has been updated successfully, "**404 Not Found**" if the resource to be updated does not exist, and "**500 Internal Server Error**" for other errors
DELETE	Delete a resource	"**200 OK**" if the resource has been deleted successfully, "**404 Not Found**" if the resource to be deleted does not exist, and "**500 Internal Server Error**" for other errors

Example:

```
POST /data/documents/balance HTTP/1.1
Content-Type: text/xml
Host: www.mydatastore.com
<?xml version="1.0" encoding="utf-8"?>
<balance date="22082014">
<Item>Sample item</Item>
<price currency="EUR">100</price>
</balance>
HTTP/1.1 201 Created
Content-Type: text/xml
Location: /data/documents/balance
```


RESTful Web Services - principles

- **Principle 4 – Resources can have multiple representations**

- A key feature of a resource is that they may be represented in a different form than the one it is stored. Thus, it can be requested or posted in different representations. As long as the specified format is supported, the REST-enabled endpoint should use it.
- In the preceding example, we posted an xml representation of a balance, but if the server supported the JSON format, the following request would have been valid as well:

- POST /data/documents/balance HTTP/1.1
- Content-Type: application/json

Example:

```
POST /data/documents/balance
HTTP/1.1
Content-Type: application/json
Host: www.mydatastore.com
{
"balance": {
"date": ""22082014"",
"Item": "Sample item",
"price": {
"-currency": "EUR",
"#text": "100“ } } }
HTTP/1.1 201 Created
Content-Type: application/json
Location: /data/documents/balance
```


innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

RESTful Web Services - principles

- **Principle 5 – Communicate Statelessly**
- Resource manipulation operations through HTTP requests should always be considered atomic.
- All modifications of a resource should be carried out within an
- HTTP request in isolation.
- After the request execution, the resource is left in a final state, which implicitly means that partial resource updates are not supported.
- You should always send the complete state of the resource.
- Back to the balance example, updating the price field of a given balance would mean posting a complete JSON document that contains all of the balance data, including the updated price field.
- Posting only the updated price is not stateless, as it implies that the application is aware that the resource has a price field, that is, it knows its state.
- The REST goals :
  - Separation of the representation and the resource
  - Visibility
  - Reliability
  - Scalability
  - Performance

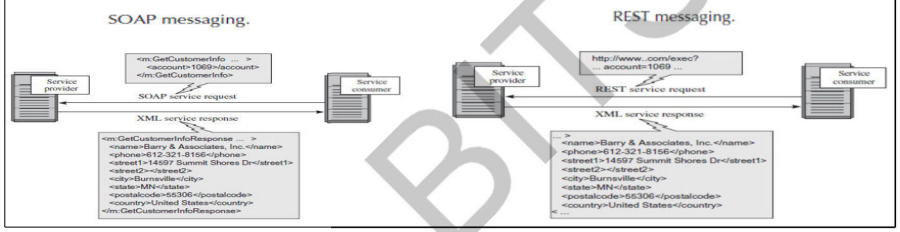
RESTful Web Services

- The acronym REST, stands for REpresentational State Transfer.
- REST is an architectural style, meaning each unique URL represents an individual object of some sort.
- A REST web service uses HTTP and supports several HTTP methods: GET, POST, PUT or DELETE.
- It also offers simple CRUD-oriented services.
- So, as you can see REST relies on a simple URL to make a request, instead of using XML.
- But in some situations, you must provide additional information, but most web services using REST rely exclusively on using the URL approach.

- Let's see an example using basic HTTP requests.
- In this case we going to use "Swagger Pet Store API" from Swagger.io (https://swagger.io/).
- Sending a GET request to /pet/{petId} would retrieve pets with a specified ID from the database.
- Sending a POST request to /pet/{petId}/uploadImage would add a new image of the pet.
- Sending a PUT request to /pet/{petId} would update the attributes of an existing pet, identified by a specified id.
- Sending a DELETE request to /pet/{petId} would delete a specified pet.

RESTful WS - Messaging

- JSON, an XML Alternative
- It is possible to use Web services without XML - JSON (JavaScript Object Notation) is one option.
- **What is JSON?**
  - JSON stands for JavaScript Object Notation
  - JSON is a lightweight data-interchange format
  - JSON is "self-describing" and easy to understand
  - JSON is language independent



RESTful WS - Messaging

- It uses name/value pairs instead of the tags used by XML.
- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

- **Why use JSON?**
- Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.
- JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects JSON.parse()
- If you receive data from a server, in JSON format, you can use it like any other JavaScript object.
- JSON Syntax Rules. JSON syntax is derived from JavaScript object notation syntax
- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

innovate achieve lead

RESTful WS - Messaging

- **Why use JSON?**
- Data is written in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value "name":"John"
- The name/value pairs in JSON provide the same type of resilience as the XML-tagged format for data exchanges.
- The name/value pairs do not have to be in any particular order to work.
- Figure also shows that XML and JSON can use the same vocabulary for the names of the data elements.

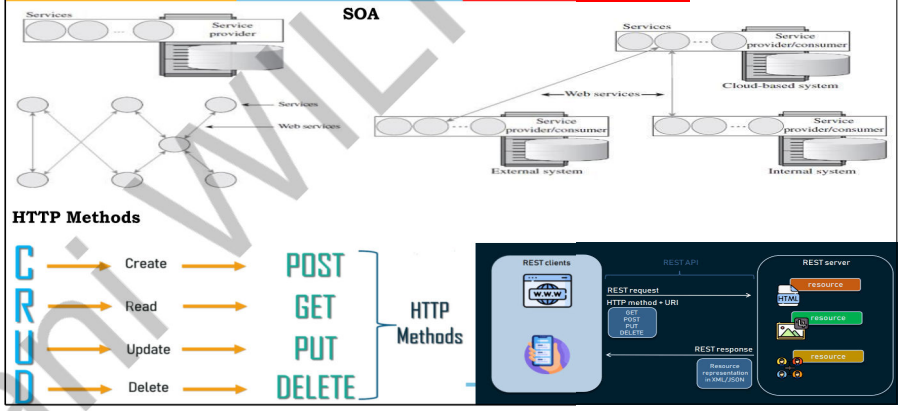
XML

```
...>
<name>Barry & Associates, Inc.</name>
<phone>612-321-8156</phone>
<street1>14597 Summit Shores Dr</street1>
<street2></street2>
<city>Burnsville</city>
<state>MN</state>
<postalcode>55306</postalcode>
<country>United States</country>
<...
```

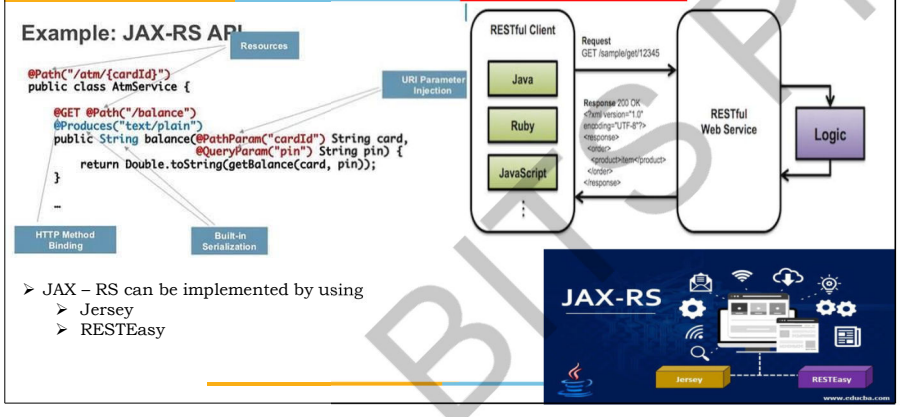
JSON

```
{
"name"       : "Barry & Associates, Inc.",
"phone"      : "612-321-8156",
"street1"    : "14597 Summit Shores Dr",
"street2"    : "",
"city"       : "Burnsville",
"state"      : "MN",
"postalcode" : "55306",
"country"    : "United States"
}
```


RESTful WS



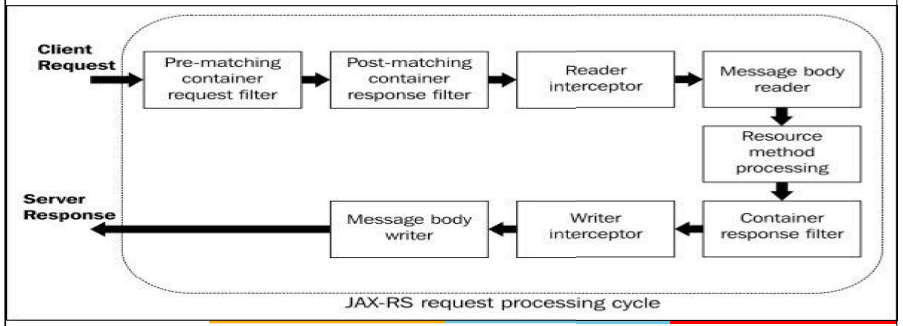
JAX - RS



- JAX – RS can be implemented by using
  - Jersey
  - RESTEasy

JAX – RS Request Processing Cycle

- The following diagram depicts the sequence of actions taking place on the server when a client invokes the JAX-RS RESTful web service



innovate achieve lead

JAX – RS Request Processing Cycle

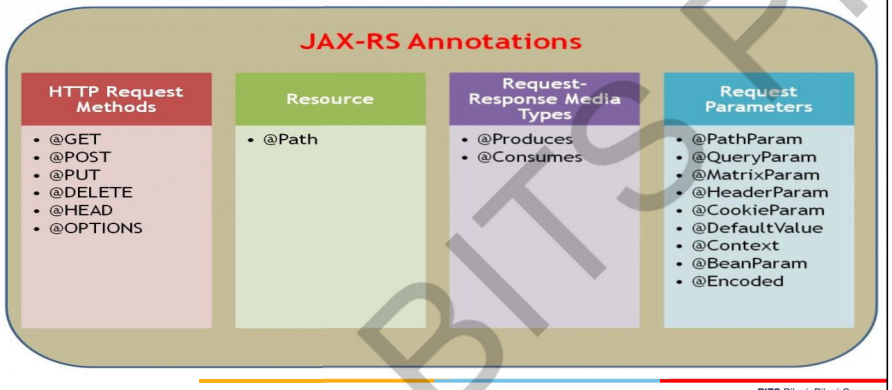
- Sequence of actions taking place on the server when a client invokes the JAX-RS RESTful web service

1. For an incoming REST API call, the container identifies the Java servlet configured for handling the REST API calls by parsing the URI and then delegates the request to the designated servlet.
2. The servlet initializes the JAX-RS runtime and kicks off the RESTful web service request processing cycle for the REST API call.
3. The JAX-RS runtime processes the filters and interceptors configured in the application in the following sequence:
    1. Runtime executes prematching filters (ContainerRequestFilter with the @Prematching annotation), which happens before resolving the resource method. The prematching filters are useful if you want to influence resource method resolution.
    2. The next step is to identify the resource method for serving the request. After the resolution of the resource method, postmatching filters are executed (filters without @Prematching). Postmatching filters are useful for performing an authentication check or for performing a basic sanity check on the request parameters.
    3. After executing all filters, the JAX-RS runtime invokes ReaderInterceptor. The reader interceptors are useful for manipulating the inbound request stream. A common use case may be to unzip the compressed stream before converting it into Java objects.
    4. The next phase in the cycle is to convert the inbound stream into the Java object representation. This is done by the matching the MessageBodyReader provider identified by the runtime.

JAX – RS Request Processing Cycle

4. After taking the incoming request through the configured filters, interceptors, and message body readers, the JAX-RS runtime invokes the resource class method identified for serving the request.
    1. If the validation fails and results in ConstraintViolationException, the runtime skips the execution of the method and identifies ExceptionMapper to handle the exception. The exception mapper generates the HTTP response content for the exception. The control flow proceeds to step 5.
    2. Upon successful validation, the resource method is executed.
    3. If the resource method throws some exception, the framework invokes the matching exception mapper to generate the HTTP response body for the exception and skips to step 5.
    4. Upon the successful execution of the method, the framework reads the response object returned by the method.
5. The runtime takes the response content through the following stages before sending it to the client:
    1. The runtime executes ContainerResponseFilter, which can be used for adding the desired response header such as cache parameters and content type.
    2. The runtime executes WriterInterceptors for the response content. The interceptor class can hold logic for manipulating the response before sending it to the client, such as zipping the response body.
    3. As the next step, the runtime invokes the appropriate MessageBodyWriter, which writes the Java type to an HTTP message to send it over HTTP. MessageBodyWriter can optionally amend or add the HTTP response headers. For instance, 200 OK is added to the response header if the method execution went well and returned an object. If the method is void, 204 No Content is set.
6. Once the response is ready, the container passes the response back to the client.

JAX – RS Annotations



Thank You

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

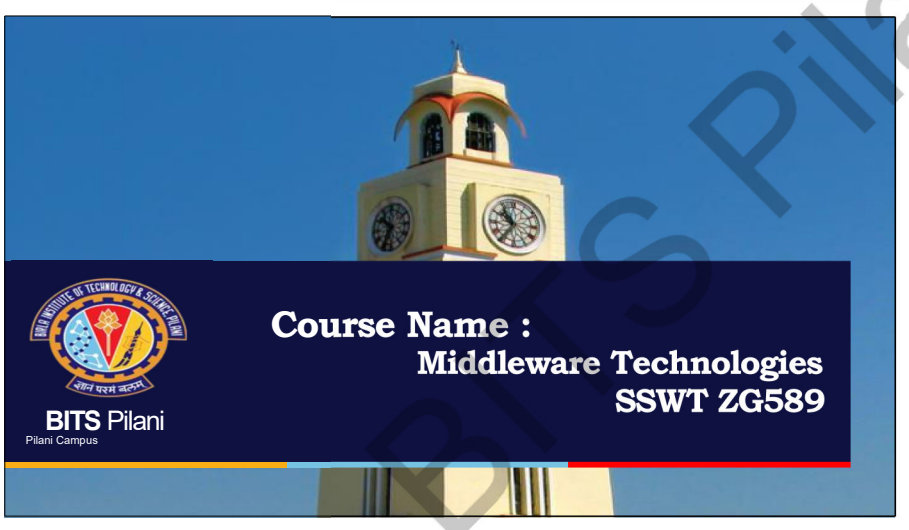
innovate achieve lead

Start Recording

innovate achieve lead

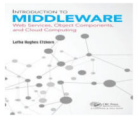
IMP Note to Students

- It is important to know that just login to the session does not guarantee the attendance.
- Once you join the session, continue till the end to consider you as present in the class.
- IMPORTANTLY, you need to make the class more interactive by responding to Professors queries in the session.
- **Whenever Professor calls your number / name ,you need to respond, otherwise it will be considered as ABSENT**



innovate achieve lead

Textbooks



T1: Letha Hughes Etzkorn - Introduction to middleware _ web services, object components, and cloud computing- Chapman and Hall_CRC (2017).

T2: William Grosso - Java RMI (Designing & Building Distributed Applications)

R1: Gregor Hohpe, Bobby Woolf - Enterprise Integration Patterns_ Designing, Building, and Deploying Messaging Solutions -Addison-Wesley Professional (2003)

R2: MongoDB in Action

Note: In order to broaden understanding of concepts as applied to Indian IT industry, students are advised to refer books of their choice and case-studies in their own organizations

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

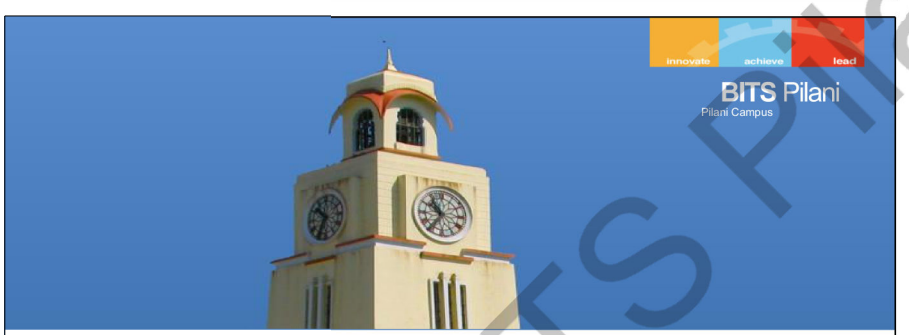
Evaluation Components

Evaluation Component	Name	Type	Weight	Duration	Schedule
EC – 1	Quiz I & II	Individual / Take-home	10%		Pre/Post Mid-Sem
EC – 2	Assignment/ Laboratory Exercises	Practical	10%		TBA
EC – 3	Mid-Semester Examination	Closed Book	30%	2 Hrs.	TBA
EC – 4	End-Semester Examination	Open Book	50%	3 Hrs.	TBA

Modular Structure

innovate achieve lead

No	Title of the Module
M1	Introduction and Evolution
M2	Enterprise Middleware
M3	Middleware Design and Patterns
M4	Middleware for Web-based Application and Cloud-based Applications
M5	Specialized Middleware



CS 12: Middleware Technologies –Services of Cloud and Cloud Providers

Agenda

innovate achieve lead

- Cloud Computing Overview
- AWS Cloud Components
- Launching EC2 for Linux / Windows
- AWS Lambda
- Hybrid Clouds
- Load Balancing
- Launhing VM in GCP

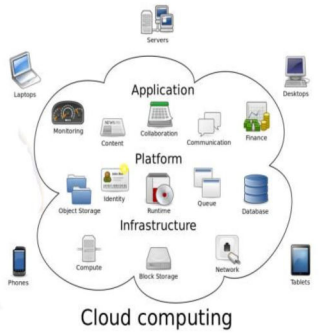
innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

innovate achieve lead

Middleware for Cloud Applications - AWS

- Cloud Computing Overview
- Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user
- Cloud computing may be available:
  - Exclusively to one organization (enterprise / private cloud)
  - Multiple organizations (public cloud)
  - Combination (hybrid)
- Cloud computing relies on sharing of resources to achieve coherence and economies of scale
- Major cloud providers:
- Amazon – AWS
- Microsoft – Azure
- Google – Google Cloud



innovate achieve lead

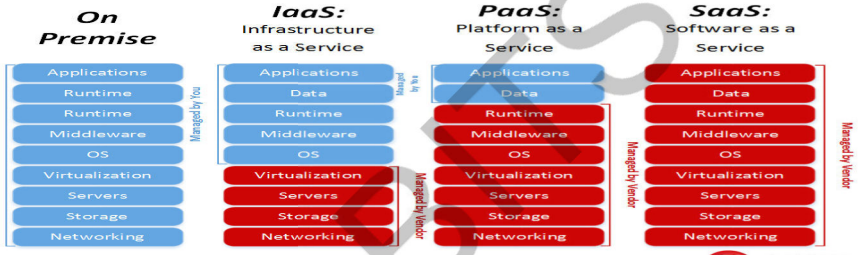
What is Cloud Computing ?

- Cloud Computing is a general term used to describe a new class of network based computing that takes place over the Internet,
  - basically a step on from Utility Computing
  - a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
  - Using the Internet for communication and transport provides hardware, software and networking services to clients
- These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).
- In addition, the platform provides on demand services, that are always on, anywhere, anytime and any place.
- Pay for use and as needed, elastic
  - scale up and down in capacity and functionalities
- The hardware and software services are available to
  - general public, enterprises, corporations and businesses markets
- Cloud computing is an umbrella term used to refer to Internet based development and services

innovate achieve lead

What is Cloud Computing ?

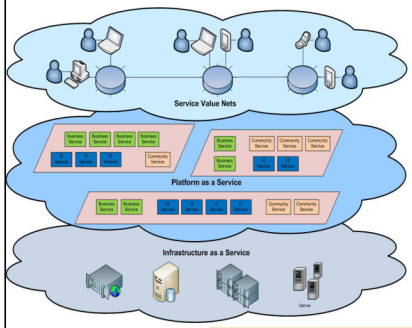
- A number of characteristics define cloud data, applications services and infrastructure:
  - **Remotely hosted:** Services or data are hosted on remote infrastructure.
  - **Ubiquitous:** Services or data are available from anywhere.
  - **Commodified:** The result is a utility computing model similar to traditional that of traditional utilities, like gas and electricity - you pay for what you would want!



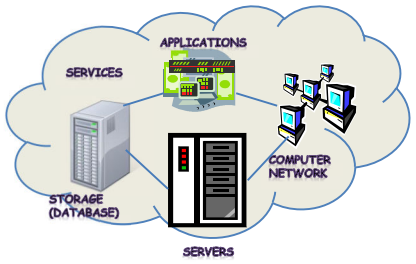
innovate achieve lead

What is Cloud Computing ?

- Cloud Architecture



- Shared pool of configurable computing resources
- On-demand network access
- Provisioned by the Service Provider



innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

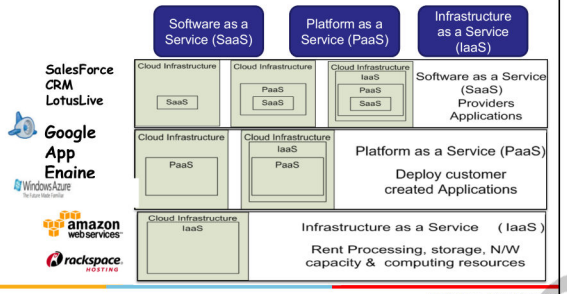
innovate achieve lead

3-4-5 rule of Cloud Computing

- NIST specifies **3-4-5 rule** of Cloud Computing
- **3** cloud service models or service types for any cloud platform
- **4** deployment models
- **5** essential characteristics of cloud computing infrastructure

- **3 Cloud Service Models**
- **1. Software as a Service (SaaS)**
- Software as a service features a complete application offered as a service on demand.
- A single instance of the software runs on the cloud and services multiple end users or client organizations.

- E.g. salesforce.com , Google Apps, Gmail, Google Drive, Dropbox,
- Cisco WebEx, SAP Concur, GoToMeeting



innovate achieve lead

3 Cloud Service Models

- **2. Platform as a service** encapsulates a layer of software and provides it as a service that can be used to build higher-level services.

- 2 Perspectives for PaaS :-
- 1. Producer:-
- Someone producing PaaS might produce a platform by integrating an OS, middleware, application software, and even a development environment that is then provided to a customer as a service.
- 2. Consumer:-
- Someone using PaaS would see an encapsulated service that is presented to them through an API.
- The customer interacts with the platform through the API, and the platform does what is necessary to manage and scale itself to provide a given level of service.
- Virtual appliances can be classified as instances of PaaS.

- **3. Infrastructure as a service** delivers basic storage and computing capabilities as standardized services over the network.
- Servers, storage systems, switches, routers , and other systems are pooled and made available to handle workloads that range from application components to high-performance computing applications.

Popular examples of PaaS include:

- AWS Elastic Beanstalk
- Windows Azure
- Heroku
- Force.com
- Google App Engine
- OpenShift

Popular examples of IaaS include:

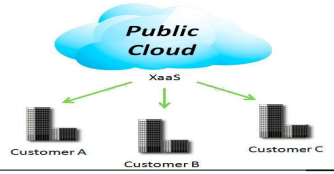
- DigitalOcean.
- Linode.
- Rackspace.
- Amazon Web Services (AWS)
- Cisco Metacloud.
- Microsoft Azure.
- Google Compute Engine (GCE)

innovate achieve lead

4 Deployment Models

- **1. Public Cloud**
- Mega-scale cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

- Types and Examples of Public Cloud Storage
- Amazon Web Services
- Microsoft Azure
- IBM Cloud
- Google Cloud Platform
- Oracle Cloud



- **2. Private Cloud**
- The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

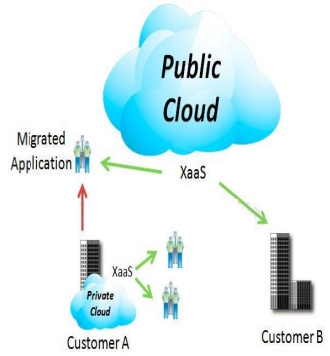
Private Cloud Company	Top Solutions
Oracle	Oracle Cloud Platform, Managed Cloud Services
IBM / Red Hat OpenShift	Red Hat OpenShift, Hosted Private Cloud Services
Microsoft	Azure Stack, Azure Private Cloud
Cisco	Cisco ONE Enterprise Cloud Suite, Cisco CloudCenter

innovate achieve lead

4 Deployment Models

- **3. Hybrid Cloud**
- The cloud infrastructure is a composition of two or more clouds (private or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability

- Top 5 Hybrid Cloud Providers:
- Amazon
- Microsoft
- Google
- Cisco
- NetApp

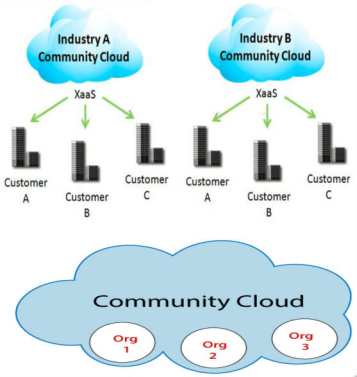


innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

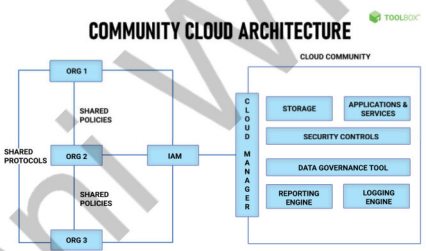
4 Deployment Models

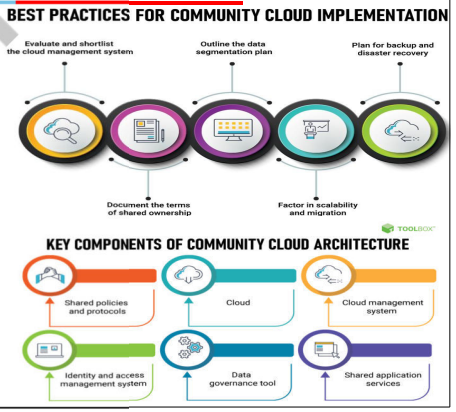
- **4. Community Cloud**
- Community Clouds are when an 'infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).
- It may be managed by the organizations or a third party and may exist on premise or off premise' according to NIST.
- A community cloud is a cloud service shared between multiple organizations with a common tie/goal/objective.
- E.g. OpenCirrus



4 Deployment Models

- **4. Community Cloud**
- Examples of Community Cloud
  - Community cloud for government sector
  - Community cloud for the healthcare sector
  - Community cloud for the education sector





5 Characteristics of Cloud Computing

5 Essential Characteristics of Cloud Computing

Ref: The NIST Definition of Cloud Computing
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Source: http://aka.ms/532

- On demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

5 Characteristics of Cloud Computing

- **Common Characteristics**

Massive Scale	Resilient Computing
Homogeneity	Geographic Distribution
Virtualization	Service Orientation
Low Cost Software	Advanced Security

- **Essential Characteristics:**

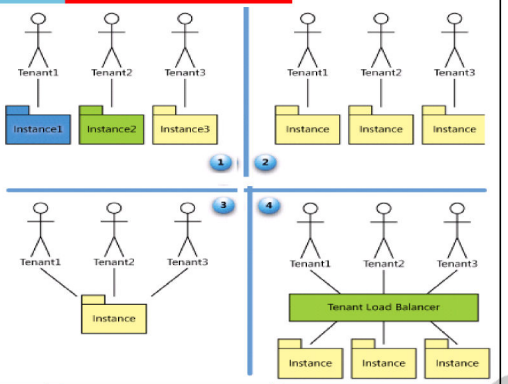
On Demand Self-Service	
Broad Network Access	Rapid Elasticity
Resource Pooling	Measured Service

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

SaaS Maturity Model

- Level 1: Ad-Hoc/Custom – One Instance per customer
- Level 2: Configurable per customer
- Level 3: configurable & Multi-Tenant-Efficient
- Level 4: Scalable, Configurable & Multi-Tenant-Efficient



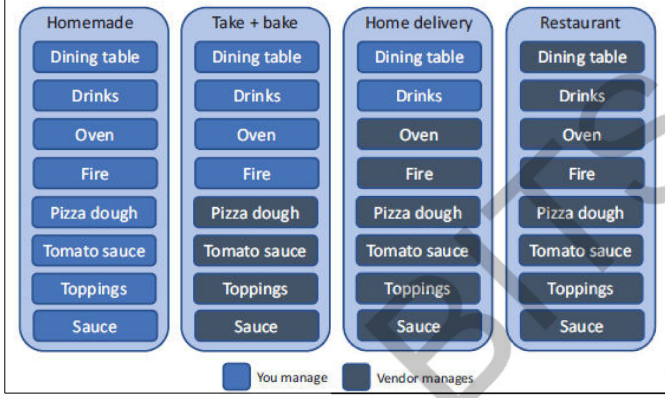
Different Cloud Computing Layers

Application Service (SaaS)	MS Live/ExchangeLabs, IBM, Google Apps; Salesforce.com, Quicken Online, Zoho, Cisco
Application Platform	Google App Engine, Mosso, Force.com, Engine Yard, Facebook, Heroku, AWS
Server Platform	3Tera, EC2, SliceHost, GoGrid, RightScale, Linode
Storage Platform	Amazon S3, Dell, Apple, …

- Cloud Computing Service Layers

	Services	Description
Application Focused	Services	Services – Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa
	Application	Application – Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online
	Development	Development – Software development platforms used to build custom cloud based applications (PAAS & SAAS) such as SalesForce
Infrastructure Focused	Platform	Platform – Cloud based platforms, typically provided using virtualization, such as Amazon EC2, Sun Grid
	Storage	Storage – Data storage or cloud based NAS such as CTERA, iDisk, CloudNAS
	Hosting	Hosting – Physical data centers such as those run by IBM, HP, NaviSite, etc.

Case Study - Pizza as a Service model



- As you move from homemade pizza, where you provide everything, to the restaurant model, where you just show up, the responsibilities and management demands change accordingly.

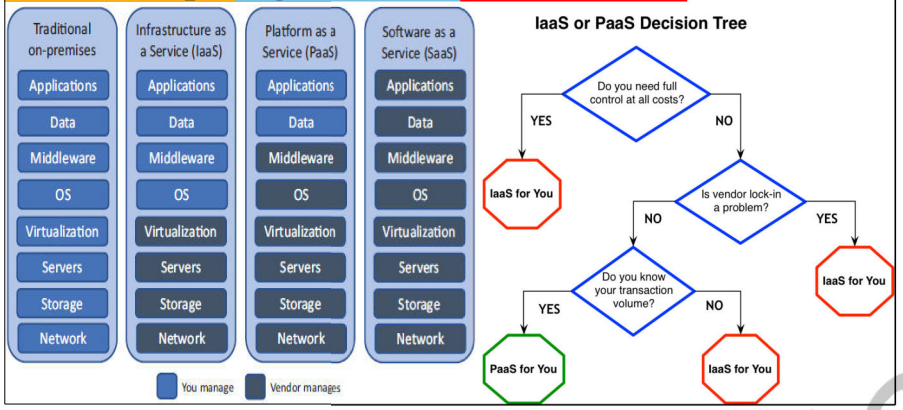
Case Study - Pizza as a Service model

- In the Pizza as a Service model, there are four options to choose from.
- As you progress through the models, you worry less and less about the process to eat a slice of pizza:

- **Homemade— On Premises**
  - You make the dough; add the sauce, toppings, and cheese;
  - Bake the pizza in your oven; get drinks; and sit down to eat at your dining table.
- **Take + bake— IaaS**
  - You buy a ready-made pizza.
  - You just need to bake it in your oven, get drinks, and sit down to eat at your dining table.
- **Home delivery— PaaS**
  - You order a pizza delivered to your home.
  - You just need to get drinks and sit down to eat at your dining table.
- **Restaurant— SaaS**
  - You want to go out and eat pizza with minimal effort!

innovate achieve lead

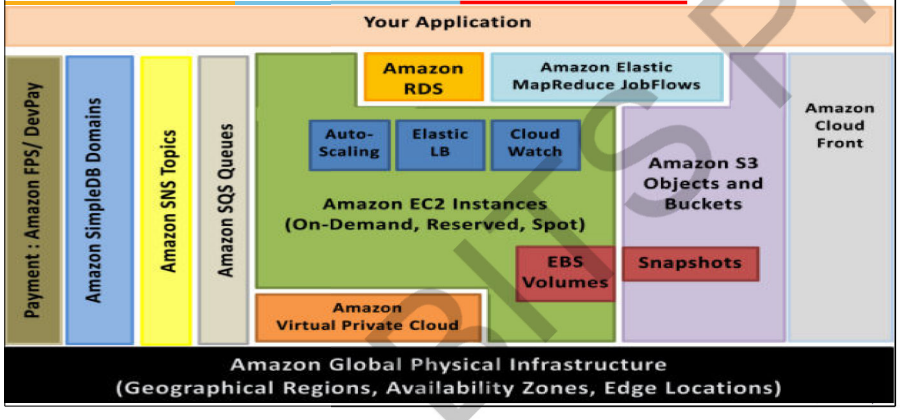
Cloud computing service model



Amazon Web Services

- Amazon Web Services Cloud
  - Provides highly reliable and scalable infrastructure for deploying web-scale solutions
  - With minimal support and administration costs
  - More flexibility than own infrastructure, either on premise or at a data center facility
- AWS – Cloud Offerings
  - Amazon Simple Storage Service (S3), which provides a highly reliable and highly available object store over HTTP
  - Amazon SimpleDB, a key-value store and
  - Amazon Relational Database Service (RDS), which provides a MySQL instance in the cloud.
  - Amazon Elastic Compute Cloud (EC2), which is Amazon's IaaS offering

AWS Infrastructure Services



Amazon Elastic Compute Cloud (Amazon EC2)

- Web service that provides resizable compute capacity in the cloud
- Can be bundled with OS, application software and associated configuration settings into an Amazon Machine Image (AMI).
- Use these AMIs to provision multiple virtualized instances
- Decommission them using simple web service calls to scale capacity up and down quickly, as capacity requirement changes.

- On-Demand Instances - pay for the instances by the hour
- Reserved Instances - pay a low, one-time payment and receive a lower usage rate to run the instance
- Spot Instances - bid for unused capacity and further reduce your cost.
- Instances can be launched in one or more geographical regions.

- Each region has multiple Availability Zones.
- Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones

innovate achieve lead

innovate achieve lead

Amazon Elastic Compute Cloud (Amazon EC2)

- The other important type of IaaS is Compute as a Service, where computing resources are offered as a service.
- Of course, for a useful compute as a service offering,
- It should be possible to associate storage with the computing service (so that the results of the computation can be made persistent).
- Virtual networking is needed as well, so that it is possible to communicate with the computing instance.
- All these together make up Infrastructure as a Service.
- Amazon's Elastic Compute Cloud (EC2), is one of the popular Compute as a Service offering.
- Amazon EC2 allows enterprises to define a virtual server, with virtual storage and virtual networking.
- As the computational needs of an enterprise can vary greatly, some applications may be compute-intensive, and other applications may stress storage.
- Certain enterprise applications may need certain software environments and other applications may need computational clusters to run efficiently.
- Networking requirements may also vary greatly.
- This diversity in the compute hardware, with automatic maintenance and ability to handle the scale, makes EC2 a unique platform.

innovate achieve lead

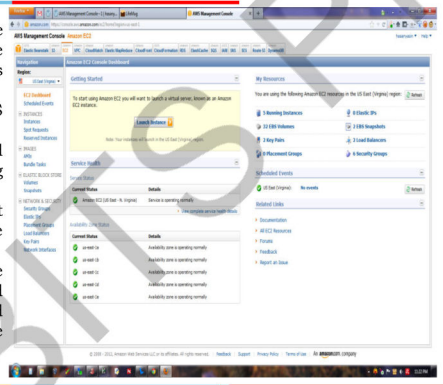
Accessing EC2 Using AWS Console

- EC2 can be accessed via the Amazon Web Services console at http://aws.amazon.com/console.
- This can be used to create an instance (a compute resource), check status of user's instances and even terminate an instance.
- Clicking on the "Launch Instance" button takes the user to the screen where a set of supported operating system images (called Amazon Machine Images, AMI) are shown to choose from.
- Once the image is chosen, the EC2 instance wizard pops up to help the user set further options for the instance, such as the specific OS kernel version to use, whether to enable monitoring using the CloudWatch tool and so on.
- Next, the user has to create at least one key-value pair that is needed to securely connect to the instance.
- Create a key-pair and save the file say my_keypair.pem in a safe place.
- The user can reuse an already created key-pair in case the user has many instances (it is analogous to using the same username-password to access many machines).
- Next, the security groups for the instance can be set to ensure the required network ports are open or blocked for the instance.
- For example, choosing the "web server" configuration will enable port 80 (the default HTTP port).
- More advanced firewall rules can be set as well.

innovate achieve lead

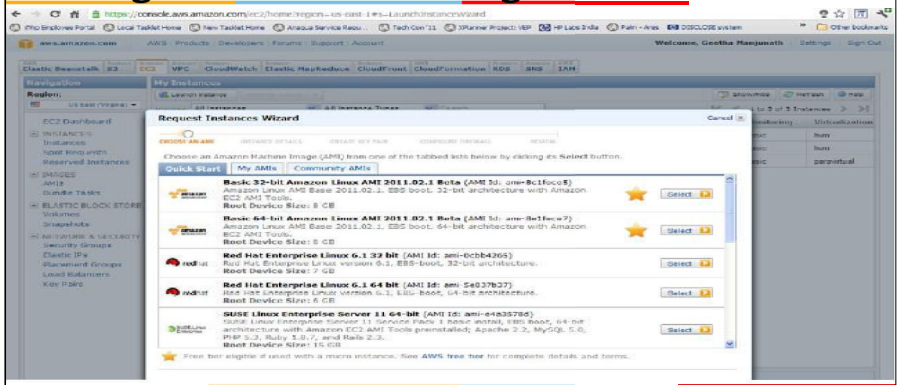
Accessing EC2 Using AWS Console

- Launching the instance gives a public DNS name that the user can use to login remotely and use as if the cloud server was on the same network as the client machine.
- Get Windows Password" button on the AWS Instance page.
- The console returns the administrator password that can be used to connect to the instance using a
- Remote Desktop application (usually available at Start-> All Programs -> Accessories -> Remote Desktop Connection).
- A description of how to use the AWS EC2 Console to request the computational, storage and networking resources needed to set up and launch a web server is described in the Simple EC2 example:
- Setting up a Web Server



innovate achieve lead

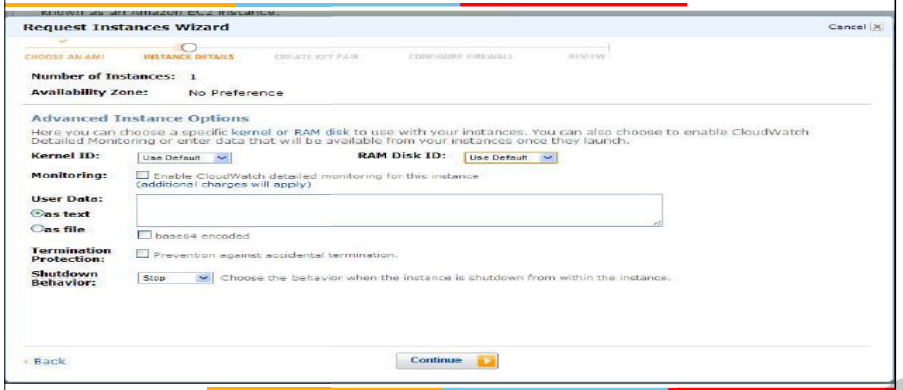
Creating an EC2 instance using the AWS console



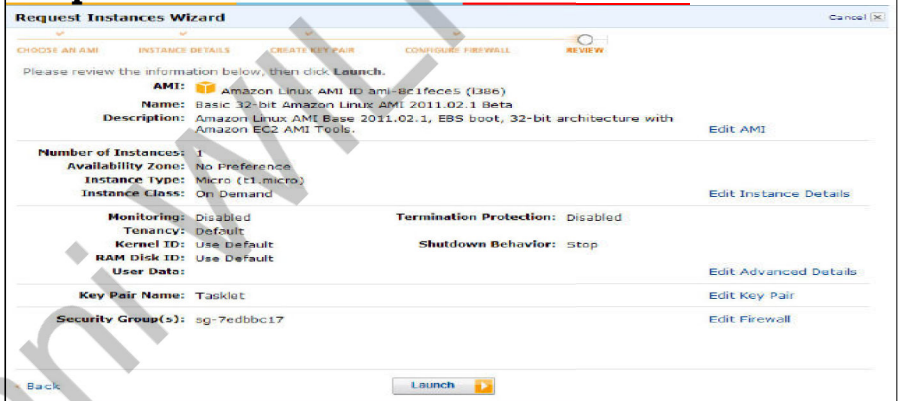
innovate achieve lead

innovate achieve lead

EC2 instance wizard



Parameters that can be enabled for a simple EC2 instance

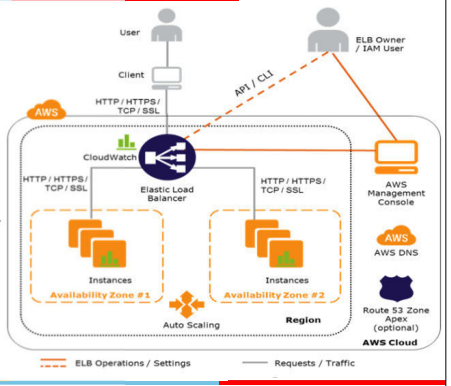


Infrastructure Services

- Elastic IP addresses - allocate a static IP address and assigned to an instance.
- CloudWatch: Enable monitoring Amazon EC2 instance - - visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic).
- Auto-scaling - to automatically scale capacity on certain conditions based on metric that Amazon CloudWatch collects.
- Elastic LB – distribute incoming traffic by creating an elastic load balancer
- Amazon Elastic Block Storage (EBS) - volumes provide network-attached persistent storage to Amazon EC2 instances.

Elastic Load Balancer (ELB)

- Load Balancing and Scaling: EC2 provides the Elastic Load Balancer, which is a service that balances the load across multiple servers.
- The default load balancing policy is to treat all requests as being independent.
- However, it is also possible to have timer-based and application controlled sessions, whereby successive requests from the same client are routed to the same server based upon time or application direction.
- The load balancer also scales the number of servers up or down depending upon the load.
- This can also be used as a failover policy, since failure of a server is detected by the Elastic Load Balancer.
- Subsequently, if the load on the remaining server is too high, the Elastic Load Balancer could start a new server instance.



innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Elastic Block Service (EBS)

- In the same way that S3 provides file storage services, EBS provides a block storage service for EC2.
- It is possible to request an EBS disk volume of a particular size and attach this volume to one or multiple EC2 instances using the instance ID returned during the time the volume is created.
- Unlike the local storage assigned during the creation of an EC2 instance, the EBS volume has an existence independent of any EC2 instance, which is critical to have persistence of data, as detailed later.

- Comparison of Instance Storage and EBS Storage

	Instance Storage	EBS storage
Creation	Created by default when an EC2 instance is created	Created independently of EC2 instances.
Sharing	Can be attached only to EC2 instance with which it is created.	Can be shared between EC2 instances.
Attachment	Attached by default to S3-backed instances; can be attached to EBS-backed instances	Not attached by default to any instance.
Persistence	Not persistent; vanishes if EC2 instance is terminated	Persistent even if EC2 instance is terminated.
S3 snapshot	Can be snapshotted to S3	Can be snapshotted to S3

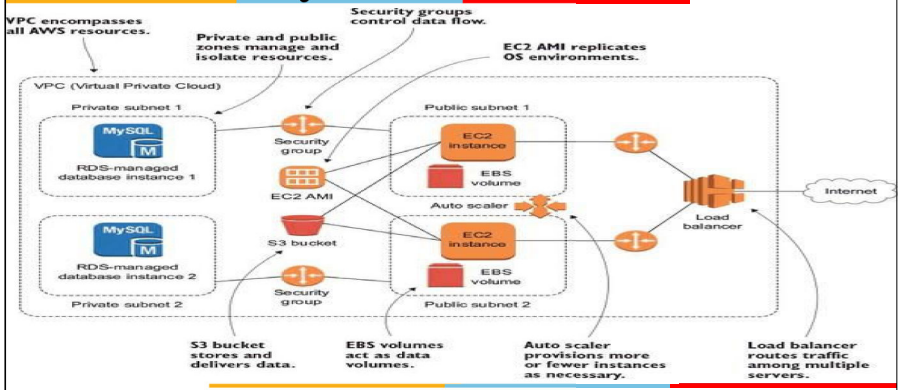
Elastic IP Addresses

- These IP addresses are independent of any instance, but are associated with a particular Amazon EC2 account and can be dynamically assigned to any instance (in which case, the public IP address is de-assigned).
- Therefore, they are useful for implementing failover.
- Upon failure of one EC2 instance, the Elastic IP address can be dynamically assigned to another EC2 instance.
- Unlike instance IP addresses, Elastic IP addresses are not automatically allocated; they have to be generated when needed.

Infrastructure Services

- Amazon S3 is highly durable and distributed data store.
- With a simple web services interface, store and retrieve large amounts of data as objects in buckets (containers) at any time, using standard HTTP
- Amazon SimpleDB - Provides the core functionality of a database, real-time lookup and simple querying of structured data
- Amazon Relational Database Service - provides an easy way to setup, operate and scale a relational database in the cloud.
- Amazon Elastic MapReduce - provides a hosted Hadoop framework
- AWS Identity and Access Management (IAM) – enables multiple User creation with unique security credentials and manage the permissions for each of these Users

AWS and Elasticity



innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

innovate achieve lead

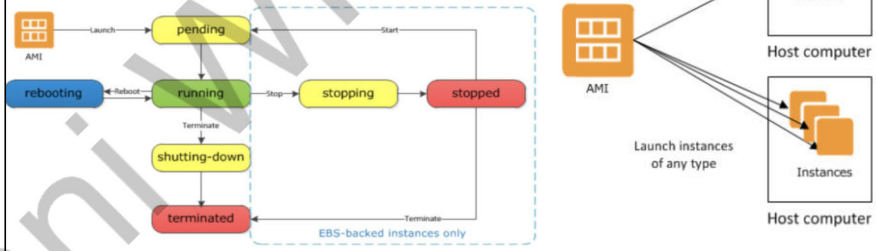
Middleware for Cloud Applications -AWS

- AWS Components
  - Compute – EC2, Elastic Beanstalk, Auto scaling
  - Storage and Content Delivery – S3, EBS
  - Database - RDS
  - Networking – VPC, ELB
  - Analytics
  - Enterprise Applications
  - Internet of Things
  - Mobile Services
  - Developer Tools
  - Management Tools
  - Security and Identity
  - Application Services
  - Software (maintenance)

innovate achieve lead

Middleware for Cloud Applications -AWS

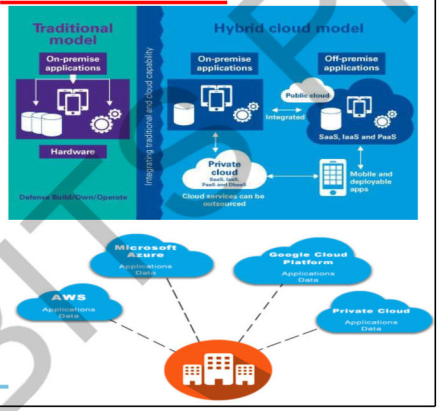
- EC2 AMI's and Instance
  - Amazon Machine Image (AMI) is the template for a software configuration (OS, application server etc.)
  - Instance(s) is a virtual server in the cloud that's a copy of AMI template



innovate achieve lead

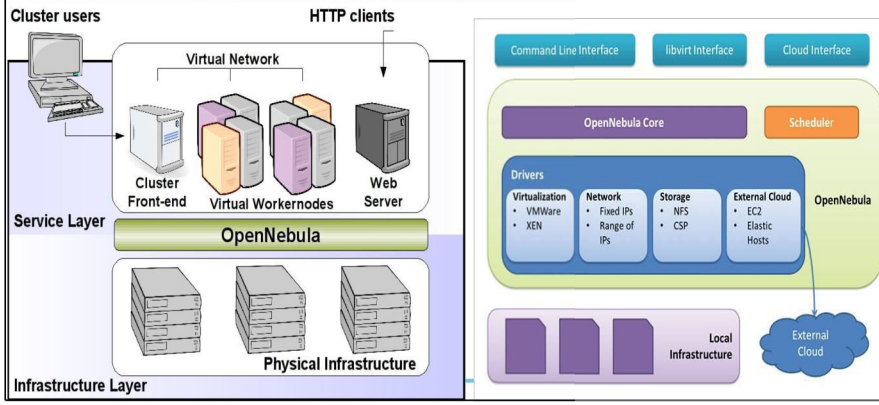
Hybrid Cloud & Multi Cloud Architectures

- **Hybrid Cloud:**
- Hybrid cloud has part of an enterprise's computing is provided by a separate cloud provider and part is done in house
- **Multi cloud** setup has an enterprise utilizing services of multiple cloud service providers in tandem (optionally with their on-prem traditional infra or private cloud)
- Advantages
  - Cost effective
  - Independence from a single provider
  - Flexibility through choice
- Disadvantages
  - Complexity of managing affairs
  - Security is a challenge
  - Redundancy and High availability across providers



innovate achieve lead

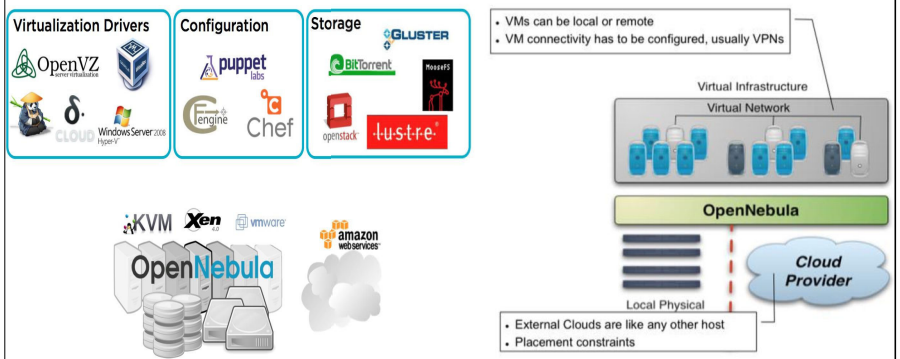
Cloud eco-system



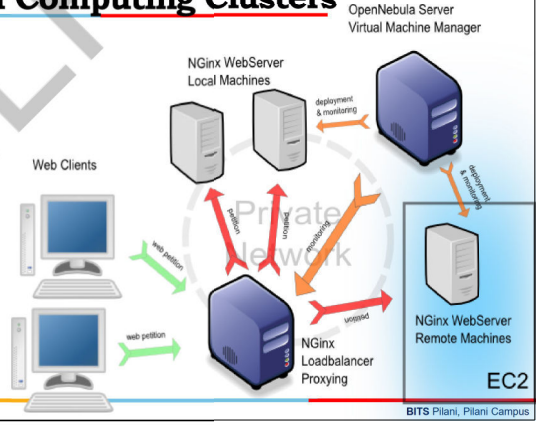
innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Cloud eco-system - OpenNebula



On-demand Scaling of Computing Clusters



- **On-demand Scaling of Web Servers**
- Elastic execution of the NGinx web server
- The capacity of the elastic web application can be dynamically increased or decreased by adding or removing NGinx instances

Middleware for Cloud Applications -AWS

- **AWS Lambda**
  - AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part
  - of AWS
  - The purpose of Lambda, as compared to AWS EC2, is to simplify building smaller, on-demand
  - applications that are responsive to events and new information
  - AWS Lambda supports securely running native Linux executables

Middleware for Cloud Applications –AWS Lambda

- AWS Lambda lets you run code without provisioning or managing servers.
- You pay only for the compute time you consume.
- With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.
- Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.
- You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.
- What is Serverless?
  - Serverless is a term that generally refers to serverless applications.
  - Serverless applications are ones that don't need any server provision and do not require to manage servers

innovate achieve lead

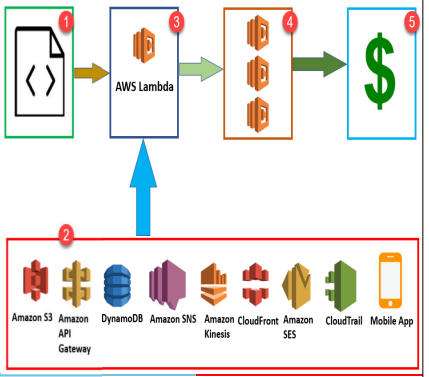
BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Middleware for Cloud Applications –AWS Lambda

- AWS LAMBDA is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services.
- Therefore you don't need to worry about which AWS resources to launch, or how will you manage them.
- Instead, you need to put the code on Lambda, and it runs.
- In AWS Lambda the code is executed based on the response of events in AWS services such as add/delete files in S3 bucket, HTTP request from Amazon API gateway, etc.
- However, a Lambda can only be used to execute background tasks.
- AWS Lambda helps you to focus on your core product and business logic instead of managing operating system (OS) access control, OS patching, right-sizing, provisioning, scaling, etc.

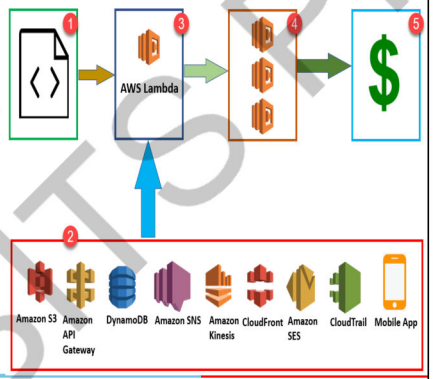
AWS Lambda – How does it work ?

- Step 1: First upload your AWS Lambda code in any language supported by AWS Lambda. Java, Python, Go, and C# are some of the languages that are supported by AWS lambda.
- Step 2: These are some AWS services which allow you to trigger AWS Lambda.
- Step 3: AWS Lambda helps you to upload code and the event details on which it should be triggered.
- Step 4: Executes AWS Lambda Code when it is triggered by AWS services:
- Step 5: AWS charges only when the AWS lambda code executes, and not otherwise.



AWS Lambda – How does it work ?

- This will happen in the following scenarios:
  - Upload files in an S3 bucket
  - When HTTP get/post endpoint URL is hit
  - For adding/modifying and deleting Dynamo DB tables
  - In the process of data streams collection
  - Push notification
  - Hosting of website
  - Email sending



AWS Lambda – Use Cases

- **Data processing:**
  - You can use AWS Lambda to execute code in response to triggers such as changes in data, shifts in system state, or actions by users.
  - Lambda can be directly triggered by AWS services such as S3, DynamoDB, Kinesis, SNS, and CloudWatch, or it can be orchestrated into workflows by AWS Step Functions.
  - This allows you to build a variety of real-time serverless data processing systems.
- **Real-time File Processing**
  - You can use Amazon S3 to trigger AWS Lambda to process data immediately after an upload.
  - For example, you can use Lambda to thumbnail images, transcode videos, index files, process logs, validate content, and aggregate and filter data in real-time.
  - The Seattle Times uses AWS Lambda to resize images for viewing on different devices such as desktop computers, tablets, and smartphones

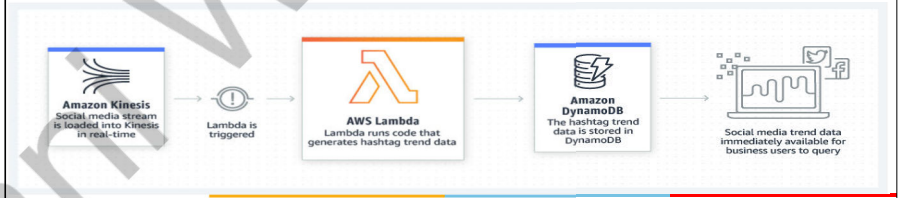
innovate achieve lead

AWS Lambda – Use Cases - The Seattle Times

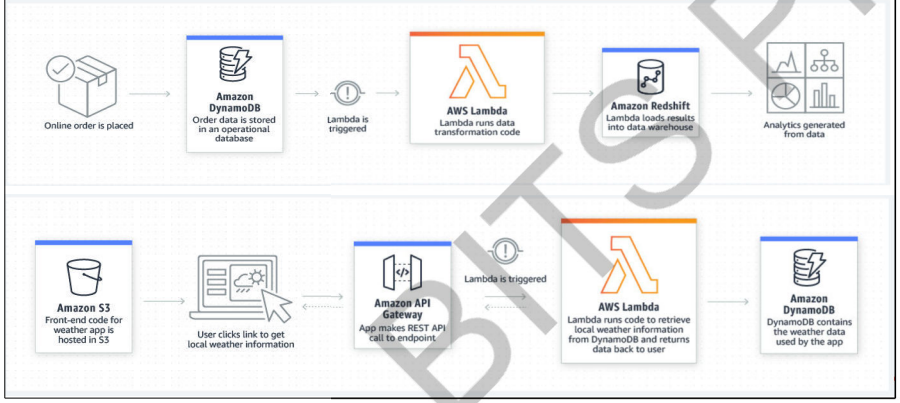
- With AWS, The Seattle Times can now automatically scale up very rapidly to accommodate spikes in website traffic when big stories break, and scale down during slower traffic periods to reduce costs. "Auto-scaling is really the clincher to this.
- "With AWS it can now serve its online readers with speed and efficiency, scaling to meet demand and delivering a better reader experience."
- News images can now be rapidly resized for different viewing environments, allowing breaking-news stories to reach readers faster.
- "AWS Lambda provides us with extremely fast image resizing,"
- "Before, if we needed an image resized in 10 different sizes, it would happen serially.
- With AWS Lambda, all 10 images get created at the same time, so it's quite a bit faster and it involves no server maintenance."

AWS Lambda – Use Cases - Localytics

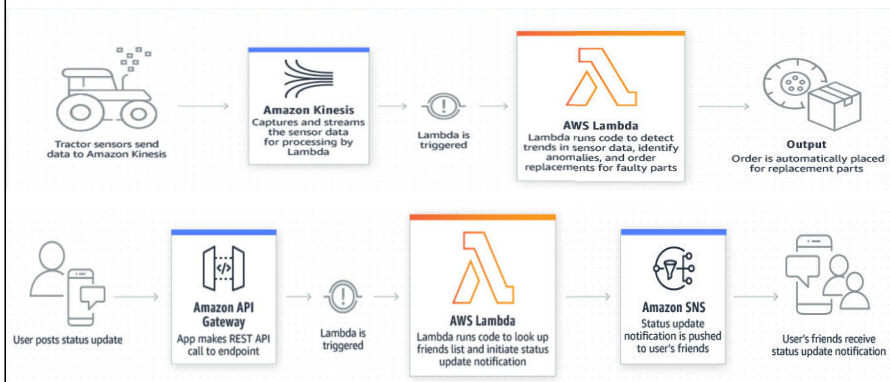
- REAL-TIME STREAM PROCESSING
- You can use AWS Lambda and Amazon Kinesis to process real-time streaming data for application activity tracking, transaction order processing, click stream analysis, data cleansing, metrics generation, log filtering, indexing, social media analysis, and IoT device data telemetry and metering.
- Localytics processes billions of data points in real-time, and uses Lambda to process historical and live data stored in S3 or streamed from Kinesis.



AWS Lambda – Use Cases - Localytics



AWS Lambda – Use Cases – IOT and Mobile



innovate achieve lead

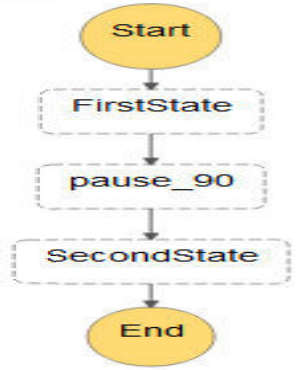
BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

AWS Lambda – Coco Cola Case Study

- The Coca-Cola Company used AWS Step Functions and other AWS services to support the Coke.com Vending Pass program.
- This program includes drink rewards earned by purchasing products at vending machines equipped to support mobile payments using the Coca-Cola Vending Pass.
- Participants swipe their NFC-enabled phones to complete an Apple Pay or Android Pay purchase, identifying themselves to the vending machine and earning credit towards future free vending purchases in the process
- After the swipe, a combination of SNS topics and AWS Lambda functions initiated a pair of calls to some existing backend code to count the vending points and update the participant's record.
- The backend code was slow to react and had some timing dependencies, leading to missing updates that had the potential to confuse Vending Pass participants.
- In order to make their solution more cost-effective, the team turned to AWS Step Functions, building a very simple state machine.
- Step Functions coordinate the components of distributed applications and microservices at scale, using visual workflows that are easy to build.
- Coke built a very simple state machine to simplify their business logic and reduce their costs.
- Step Function features such as sequential and parallel execution and the ability to make decisions and choose alternate states.

AWS Lambda – Coco Cola Case Study

- The Coke state machine looks like this:
- The FirstState and the SecondState states (Task states) call the appropriate Lambda functions while Step Functions implements the 90 second delay (a Wait state).
- This modification simplified their logic and reduced their costs.



AWS Lambda – Coco Cola Case Study

Google Cloud Platform and Creating Your First VM in GCP

- The **Google Cloud Platform** is a collection of physical and virtual assets, including computers, hard drives, and CPUs, that are housed in Google Cloud data centres.
- Each data centre is located in a worldwide region, which is comprised of zones. Within the same territory, each zone is segregated from the others.
- This distribution gives us several benefits, such as data redundancy in the event of a zone failure, lower latency by placing resources closer to the client, and more control over resources and how they may be used to improve service and lower costs.

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

Google Cloud Platform and Creating Your First VM in GCP

innovate achieve lead

- Because this is a cloud platform, you won't have to bother about purchasing and building hardware or data centres.
- You may use all of Google's hardware and software as services. These services in Google Cloud Platform give you access to the underlying hardware.
- When you begin working, you will find that your product is simply a mix of Google services and your code.

What is Virtual Machine and Google Compute Engine (GCE) ?

innovate achieve lead

When you deploy your app, you need servers. In the cloud environment, you need virtual servers to deploy and run your app. **Google Compute Engine (GCE)** will provision and maintain the virtual machines.

Virtual machines on Google Computing Engine deliver on-demand, high-scale performance and value, allowing you to quickly deploy massive compute clusters on Google's infrastructure using Windows Server or Red Hat Enterprise Linux or any Operating system of your choice.

innovate achieve lead

What is Google Compute Engine?

You need Servers to deploy applications.
Similarly, in the cloud, you need **Virtual Servers** to deploy applications.

Google Compute Engine will provide you with the **Virtual Servers** you need.

innovate achieve lead

What is Google Compute Engine?

In cloud environment, to deploy your applications, you need -

- Rent Virtual Servers
- In **Google Cloud Platform (GCP)**, Virtual Machines are the Virtual Servers.
- **Google Cloud Engine (GCE)** provisions and manages Virtual Machines.

innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

innovate achieve lead

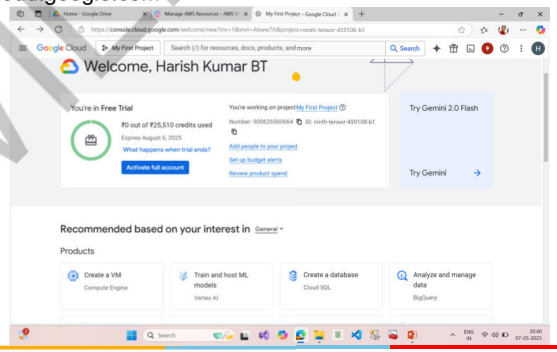
Features of Google Compute Engine

- Create and manage the lifecycle of Virtual Machine (VM) instances
- Choose the hardware of the Virtual Machine (VM) instance
- Attach storage and network policies with Virtual Machine (VM) instances
- Load Balance and Auto Scaling while dealing with multiple VM instances

innovate achieve lead

How to Open Google Cloud?

- Visit https://console.cloud.google.com

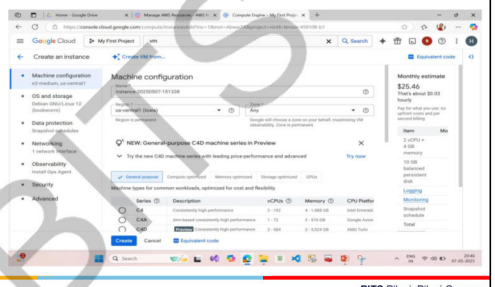


innovate achieve lead

Steps to create Virtual Machine in Google Cloud Platform (GCP)

Step 1

Inside the Google Cloud console, search for virtual machine by typing "vm" and then choose VM Instances. In the next page, click on the "**Create Instance**" button.



Create Machine configuration.

Select Low cost instance E2 Series

innovate achieve lead

Step 2

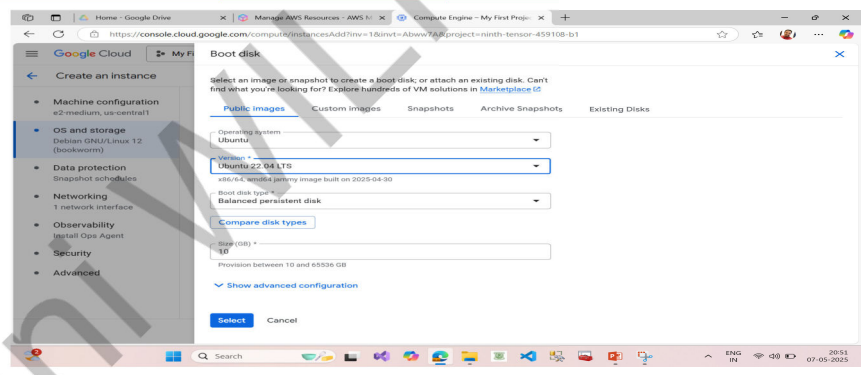
Now, we need to configure our virtual machine. Below are the basic configuration to start with

- **Name:** Place to put a name to our virtual machine
- **Labels:** It's a key/value pair to indicate different environments, services, teams and so on. Few examples –
  - environment: dev
  - app: service
  - bu: bank

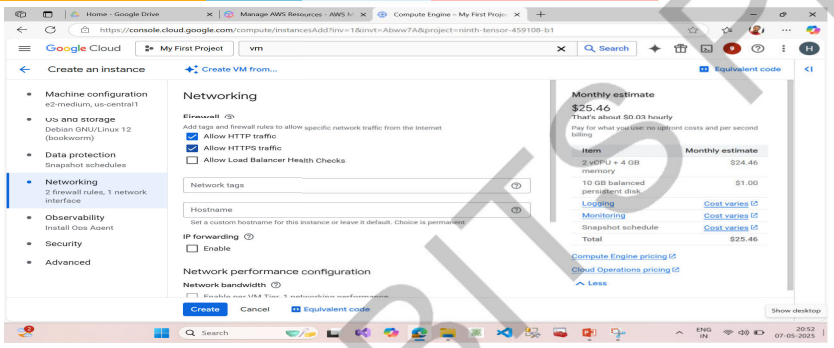
innovate achieve lead

- **Regions:** This is the place to choose the region and zones where you want to deploy your application. To understand the concept of Regions and Zones, please read my previous blog post here
- **Machine configuration:** This time we need to select the type of machine we need. GCP provides three different types of machine configuration based on the need. Pricing and discount will also change based on the machine type you choose.
  - **General Purpose**: This is mainly used for common workloads, optimized for cost and flexibility.
  - **Compute Optimised**: These are high-performance machine types used for compute-intensive workloads.
  - **Memory Optimised**: These are large-memory machine types used for memory-intensive workloads.
- **Boot disk:** Here we will select the type of operating select we want for our virtual machine. Google Cloud provides different types of public images to select. For example: Debian, Red Hat, CentOS, Ubuntu, Windows Server etc.

Select OS



Allow HTTP and HTTPS traffic



- **Identity and API access:** This time we will select the service account and level of API access. Any applications which will run on VM uses the service account to call Google Cloud APIs. For simplicity, let's select below two
  - Allow HTTP traffic
  - Allow HTTPS traffic
- **Networking:** Under this, we can select the network, subnetwork, primary internal IP, external IP etc.
- **Disks:** Here we can add additional disk/storage in our virtual machine with additional cost.

There are a couple of more configurations we can set on this page, but for simplicity, I just select the basic one to start with.

Finally, we need to click on "Create" button

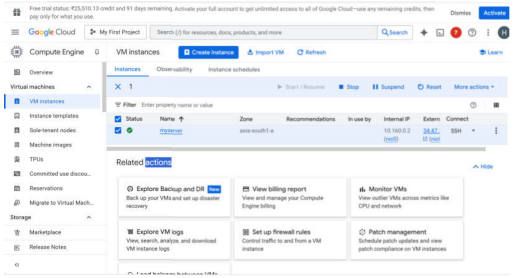
innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

innovate achieve lead

Finally

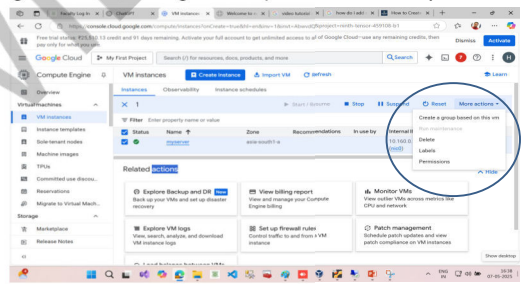
Our first Virtual Machine is up and running in Google Cloud Platform.



Start, Stop, Delete Virtual Machine in Google Cloud Platform (GCP)

innovate achieve lead

Once we have our **Virtual Machine** ready, we can go to the console to view it. For each virtual machine, there are three vertical dots(as shown in the picture below) clicking on that we will get the option to start, stop or delete our virtual machine.

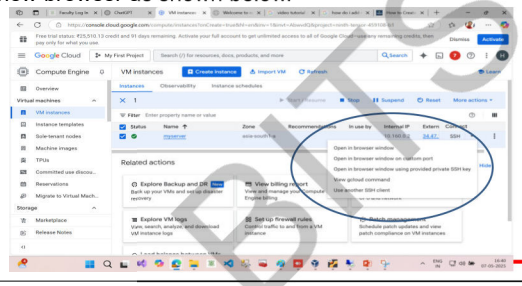


Login into Virtual Machine in Google Cloud Platform (GCP)

innovate achieve lead

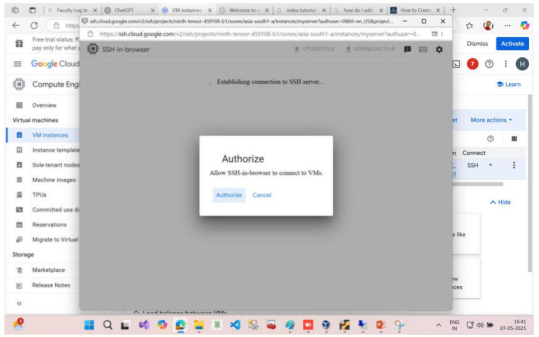
Open in browser window

We can log in to our virtual machine by clicking on the option "**Open in browser window**" from the SSH dropdown. It will open the virtual machine console in a new browser as shown below.



Click on Authorize.

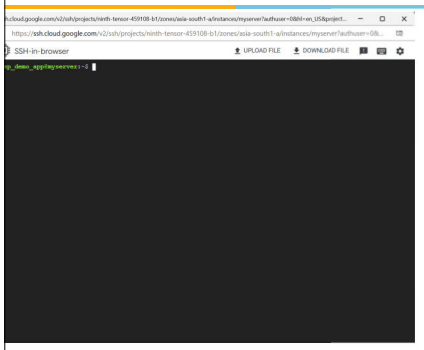
innovate achieve lead



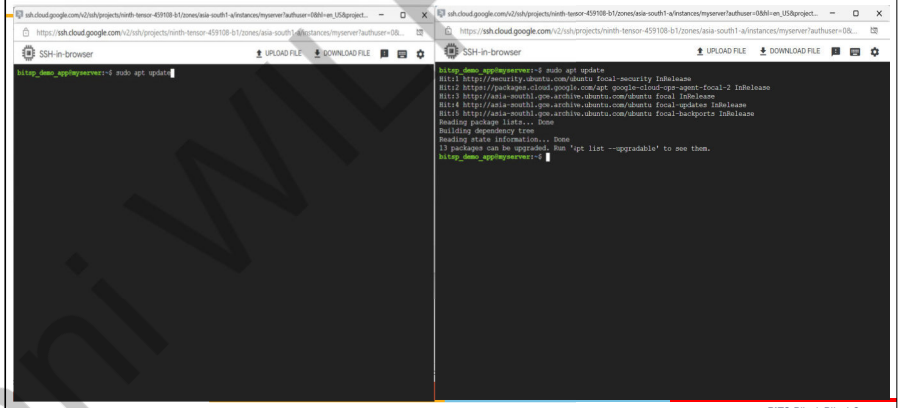
innovate achieve lead

BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai
WORK INTEGRATED LEARNING PROGRAMMES

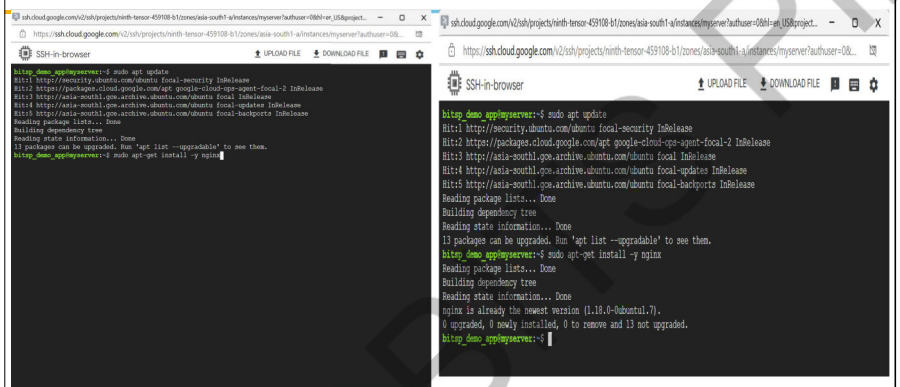
VM Window



Update apt

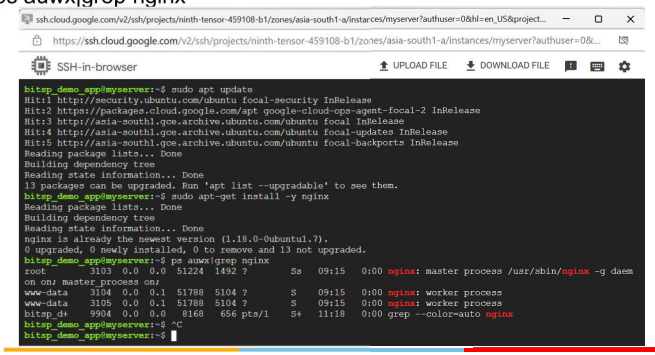


Install nginx server in your VM



Verify nginx web server is running

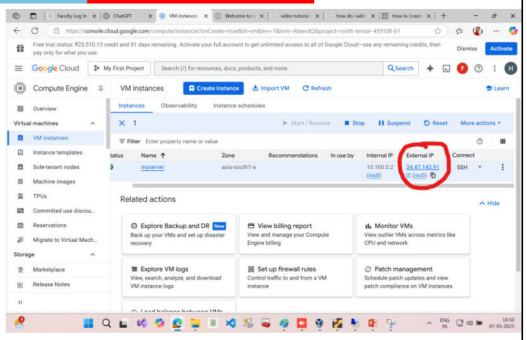
Run command: ps auwx|grep nginx



Check Web page

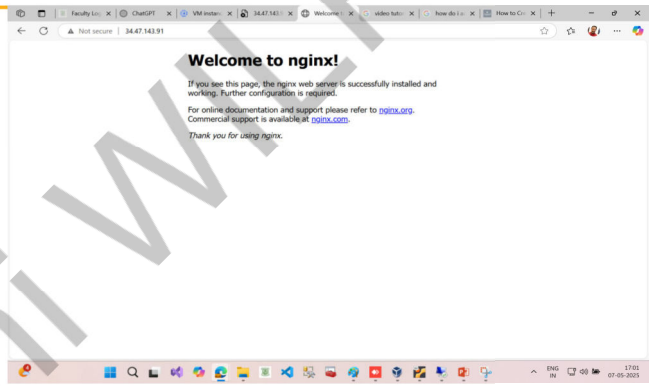
innovate achieve lead

Note External IP, refer snapshot
Open browser:
Goto url: http://34.47.143.91



You should see the default page Of nginx

innovate achieve lead



Thank You

innovate achieve lead

innovate achieve lead