

Introduction to Java EE and Enterprise Applications

Enterprise Application Development

Sachit Talagala

Principal Software Engineer/Tech Lead

IFS

Course Outline

Week	Current Topics	Lecturer Name	Date (Tentative)
1	Introduction to EAD and J2EE	Sachit Talagala	31 st July 2022
2	ERP and Big Data	Supul Jayawardena	7 th August 2022
3	NoSQL in Enterprise Application Development	Pramuditha Rathnayaka	14 th August 2022
4	Overview of Cloud Computing	Kasun Madusanka	21 st August 2022
5	Using IoT in Enterprise Applications	Tiny Pahattuge	28 th August 2022
6	Web Services in Enterprise Applications	Isuru Wijeratna	04 th September 2022
7	Enterprise Mobile Application Development – Assignment	Anjula Priyanath	11 th September 2022
8	Microservices in Enterprise Applications – Part 01	Thilina Welivita	18 th September 2022
9	Microservices in Enterprise Applications Part 2 & DevOps	Sameera Madushanka	25 th September 2022
10	EAD Methodologies	Dinushani Wickramarachchi	02 nd October 2022
11	OData in Enterprise Applications	Tharindu Delgolla	09 th October 2022
12	Enterprise Application Integration	Charmy Weerakoon	16 th October 2022

Outline

- Enterprise Applications
- What is Java EE
- EJBs
- A brief technical look at EJBs
- Java EE APIs
- Additional Section : Enterprise Application Integration.

Enterprise Applications

- Applications designed to solve problems encountered by large enterprises
- Used by large corporations, governments, etc.
- Useful for even small organizations in an increasingly networked world

Large-scale • Scalable
Reliable • Secure

Enterp. Applications Requirements

- **Large :**A multi-user, multi-developer, multi-machine, multi-component application that can manipulate massive data and utilize extensive parallel processing, network distributed resources, and complex logic. It can be deployed across multiple platforms and inter-operate with many other applications, and it is long lived.
- **Business Oriented:** Its purpose is to meet specific business requirements. It encodes business policies, processes, rules, and entities, is developed in a business organization, and is deployed in a manner responsive to business needs.
- **Mission Critical:** An enterprise application must be robust enough to sustain continuous operation. It must be extremely flexible for scalability and deployment, and allow for efficient maintenance, monitoring, and administration

What is Scalability?

- The capacity to be changed in size or scale.
- Doing the same thing in a bigger way : more users, more domains etc.
- Vertical Scalability
 - Adding resource within the same logical unit to increase capacity. e.g.: Add more server resources
- Horizontal Scalability
 - Adding multiple logical units of resources and making them work as a single unit. Clustering Distributed systems etc.

N-Tier/Multi-Tier Architecture

- N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated. That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity. The “N” in the name n-tier architecture refers to any number from 1

N-Tier/Multi-Tier Architecture

- Functionality of application separated into isolated functional areas called ‘tiers’
- 3-tiered architecture:
 - **Client tier** - client program makes requests to middle tier
 - **Middle tier** - handles client requests by processing application data (business logic – business tier)
 - **Data tier** - persistent data store (database / legacy system / etc.)

Multi-Tiered Architecture

Presentation tier

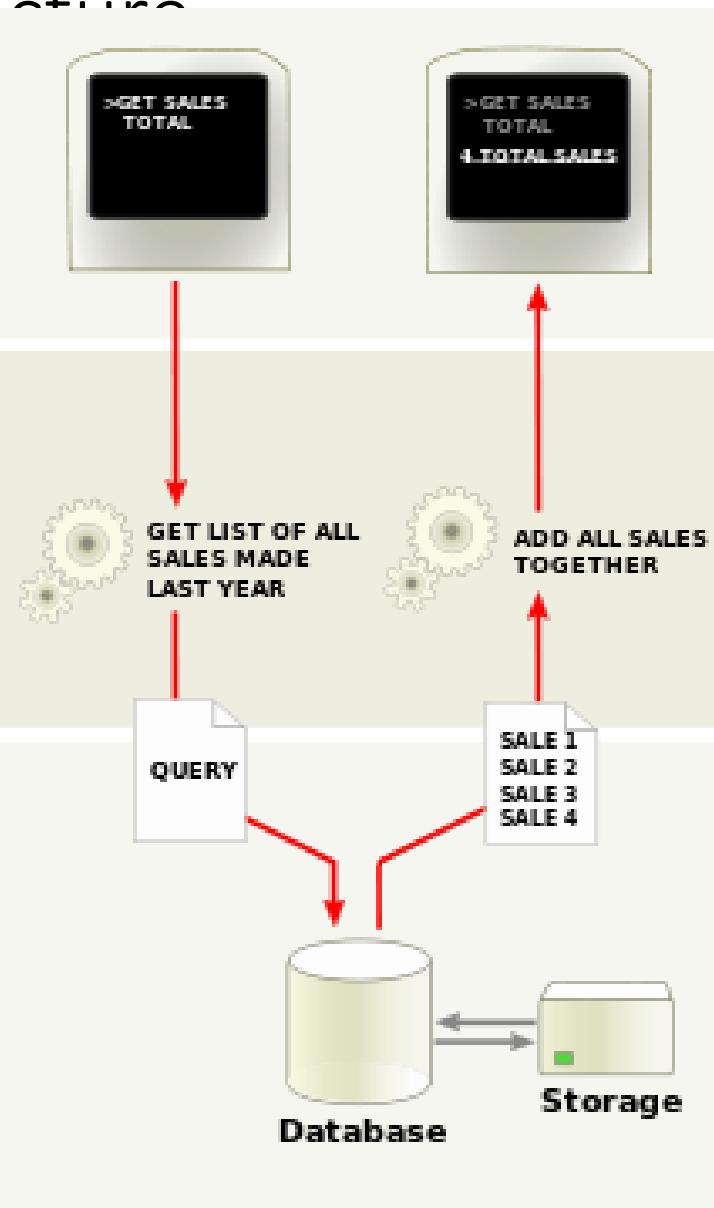
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

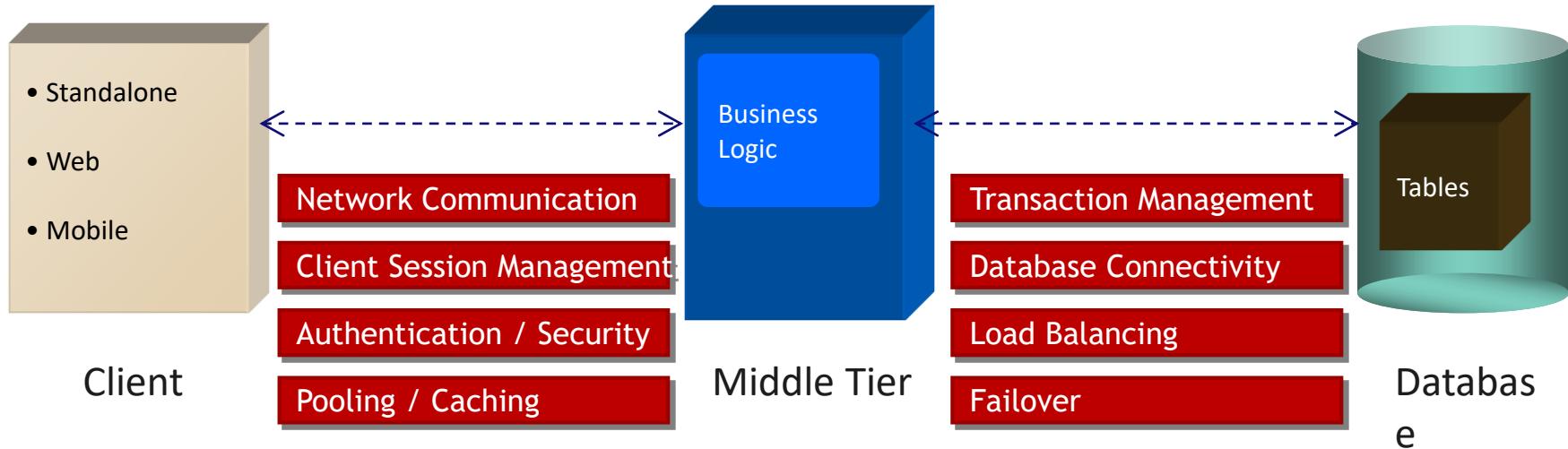
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



Benefits of N-Tier Architecture

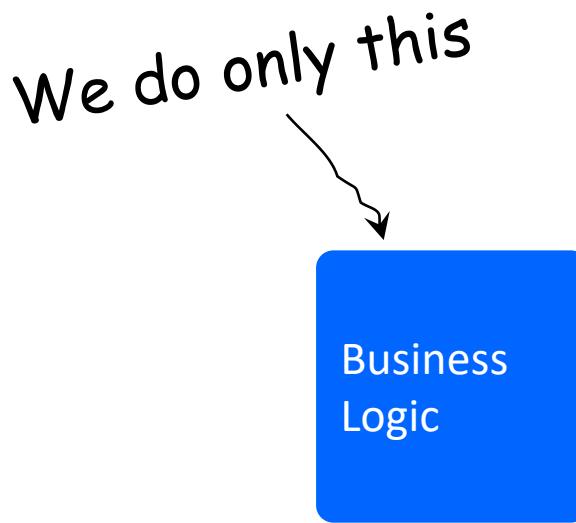
- **Secure:** You can secure each of the three tiers separately using different methods.
- **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
- **Flexible:** Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.

Enterprise Applications

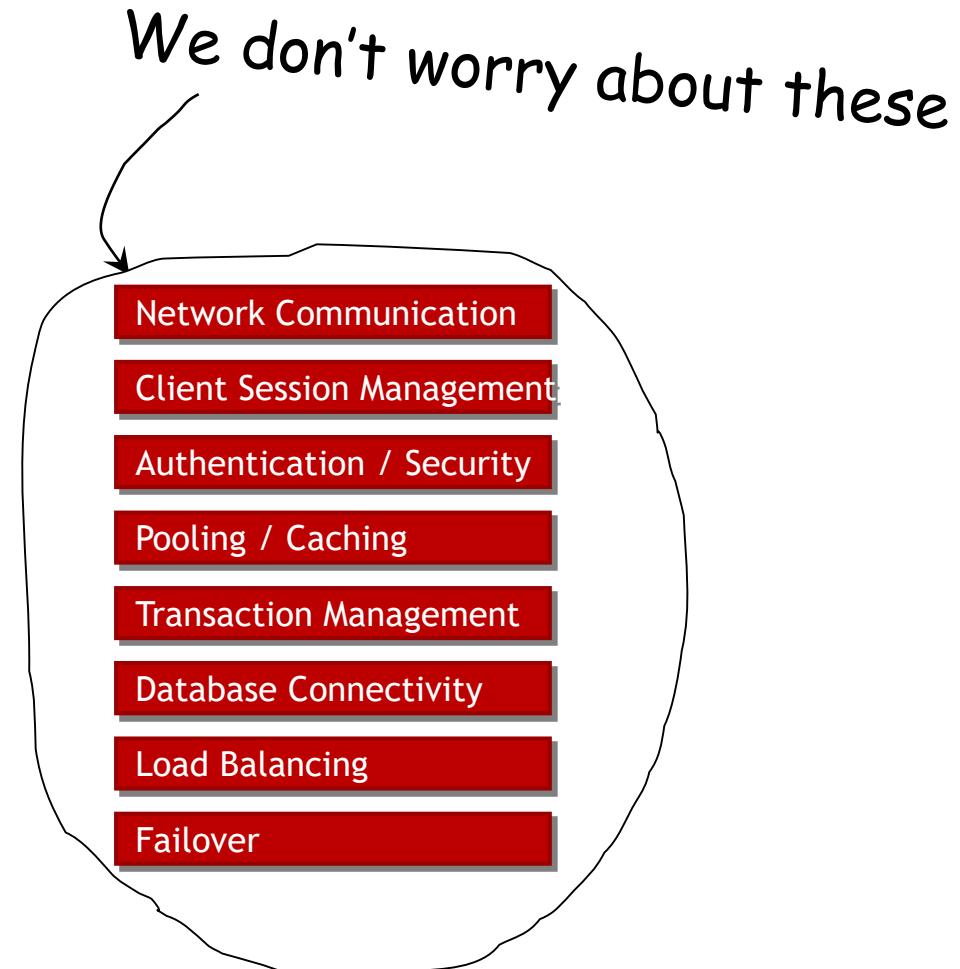


- In addition to business logic, we have to implement the other middle-tier services ('middleware')
 - Very Complex
 - Needs technical expertise
 - Time consuming
 - Very costly

Enterprise Applications



Java EE™



The Java Programming Language Platforms

There are four platforms of the Java programming language:

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
- JavaFX



Java Enterprise Edition

- The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

Jakarta EE



Jakarta EE

- After 20 years Java EE was rebranded in September 2019
- Migrated from Oracle to Eclipse foundation under Eclipse Enterprise for Java EEJ4 project
- Renaming due to licensing and legal formalities – oracle owns the Java brand name
- High focus on the cloud
- New Era in Java Eco system
- Still an extension of Java EE 8

Java EE history

- J2EE 1.2 (December 12, 1999)
- J2EE 1.3 (September 24, 2001)
- J2EE 1.4 (November 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (December 10, 2009)
- Java EE 7 (May 28, 2013)
- Java EE 8 (August 31, 2017)
- Jakarta EE8 (September 10, 2019)
- Jakarta EE9 (December 8, 2020)
- Jakarta EE9.1(May5, 2021)
- Jakarta EE10 (March 31, 2022)

Is Java/Jakarta EE relevant still?

- Mature framework for web services/API's
- Distributed systems are very common today
- Utilize existing implementations – bridges the old and new
- Supports both modern paradigms and traditional applications

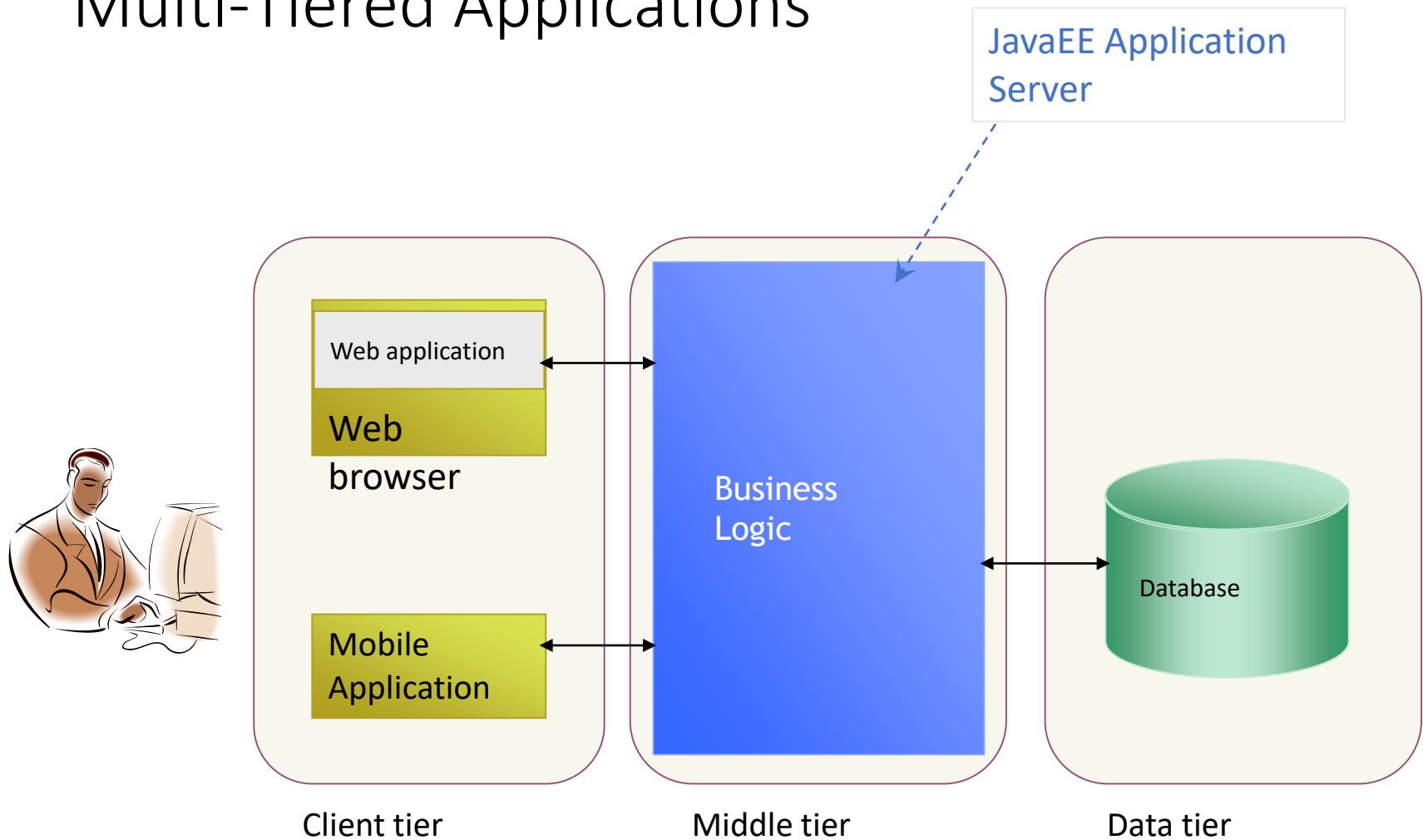
Java Technology – Java EE

Java EE

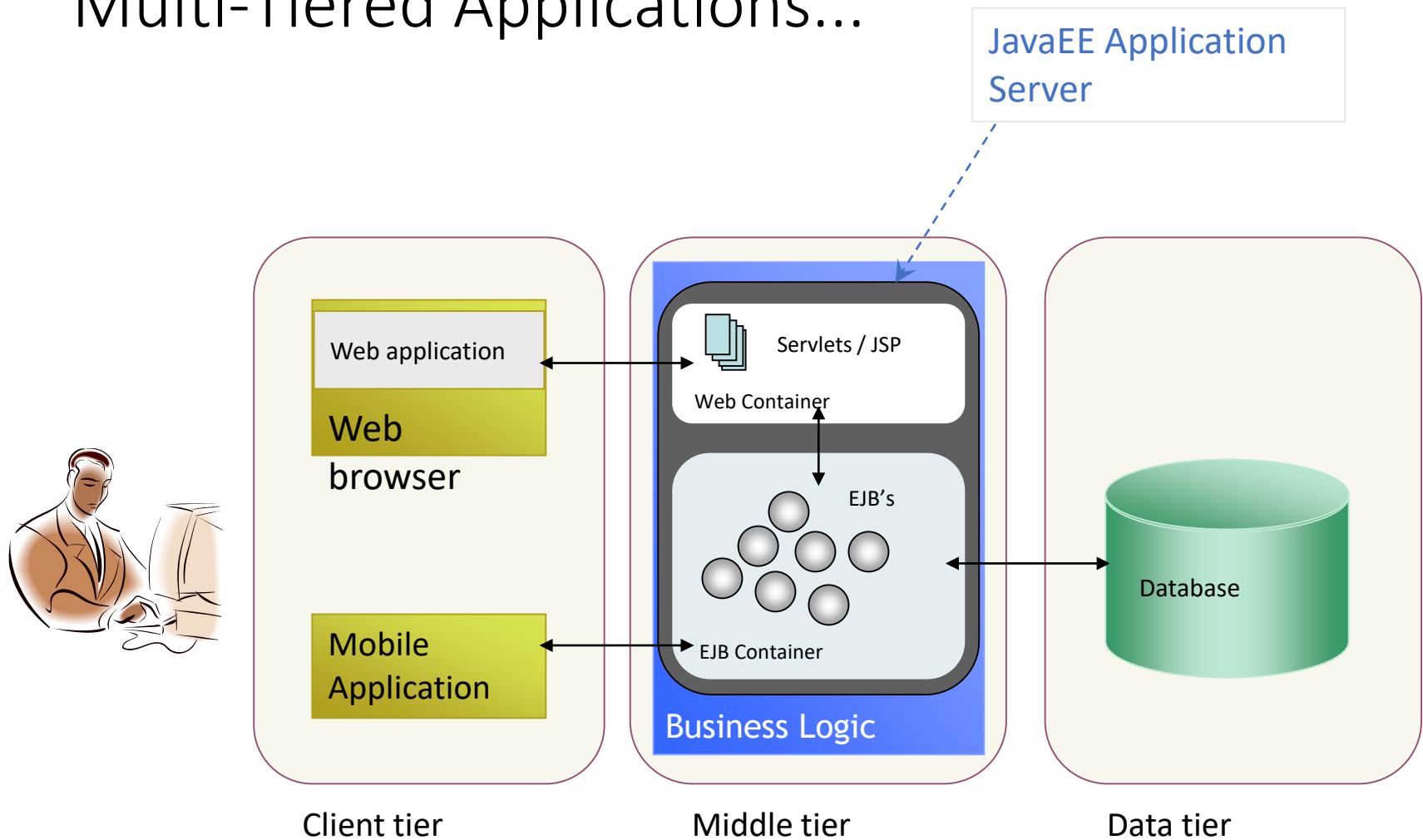
Java SE

	<p>API For developing and running large-scale, multi-tiered, scalable, reliable and secure enterprise applications</p>	<p>Development Model</p> <hr/> <p>Runtime environments (“containers”)</p>
	<p>API Provides core functionality of Java</p> <ul style="list-style-type: none">■ Basic types and objects■ Input/output■ Networking■ Security■ GUI development■ Database access■ XML processing	<p>Development tools, Deployment Technologies, other class libraries</p>
	<p>JVM</p>	

Multi-Tiered Applications



Multi-Tiered Applications...



EJB - Enterprise Java Beans

- An enterprise Java bean (EJB) is a server-side component that encapsulates the business logic of an application.
- The business logic is the code that fulfills the purpose of the application.
- Does not perform display of data (presentation) or perform database operations (persistence)

J2EE EJB Server / Container

- Manages the Enterprise beans contained within itself
- Responsibilities of the Server/Container:
 - Registering objects
 - Providing remote interfaces for the object
 - Creating and destroying object instances
 - Checking security for the object.
 - Managing active states for the object.
 - Co-ordinating distributed transactions.

Types of EJB's

- Session Beans
 - Stateful
 - Stateless
 - Singleton
- Message Driven Beans

Session beans

- Session bean encapsulates business logic only, it can be invoked by local, remote and webservice client.
- It can be used for calculations, database access etc.
- The life cycle of session bean is maintained by the application server (EJB Container).

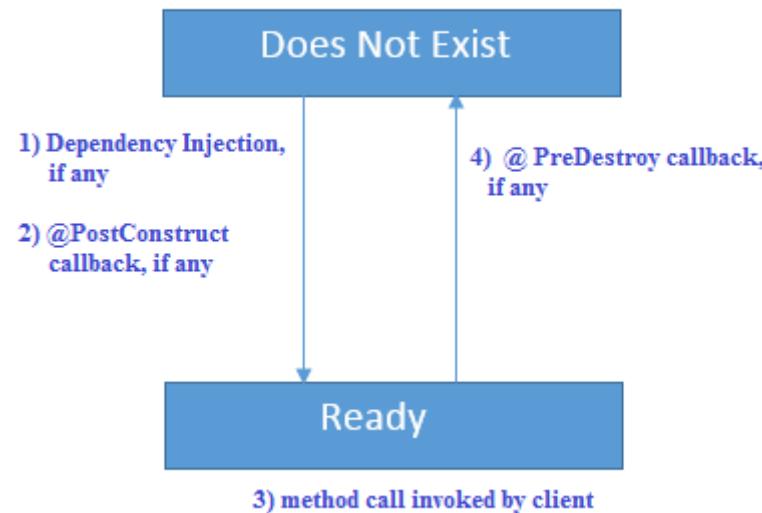
Types of Session Bean

- **1) Stateless Session Bean:** It doesn't maintain state of a client between multiple method calls.
- **2) Stateful Session Bean:** It maintains state of a client across multiple requests.
- **3) Singleton Session Bean:** One instance per application, it is shared between clients and supports concurrent access

Stateless Session Bean

- Stateless Session bean is a business object that represents business logic only. It doesn't have state (data).
- In other words, conversational state between multiple method calls is not maintained by the container in case of stateless session bean.
- The stateless bean objects are pooled by the EJB container to service the request on demand.
- It can be accessed by one client at a time. In case of concurrent access, EJB container routes each request to different instance.

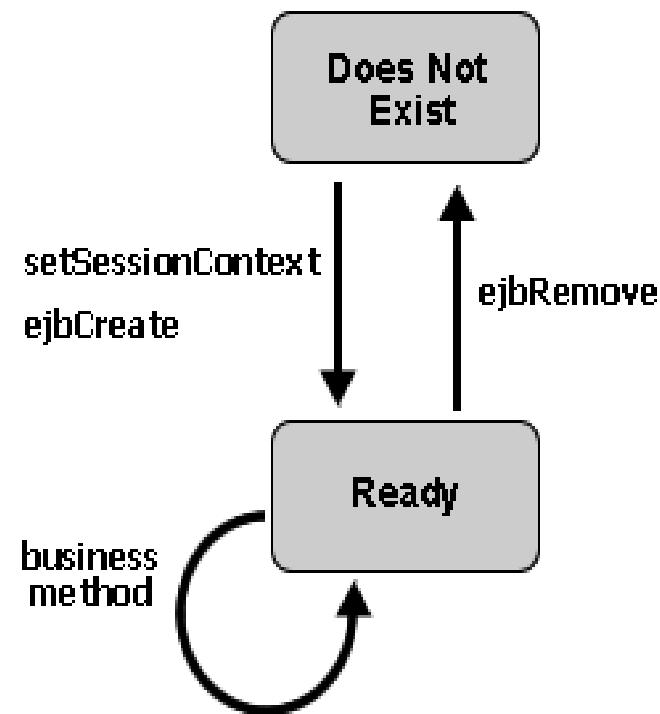
Stateless Session Bean



Stateful Session Bean

- **Stateful Session bean** is a business object that represents business logic like stateless session bean. But, it maintains state (data).
- A Stateful Session Bean maintains the conversational state with the client it is communicating.
- Each client creates a new instance of Stateful Bean and is not shared with other clients.
- When the communication between the client and bean ends, the Session Bean also terminates.

Stateful Session Bean



Singleton Session Bean

- A Singleton Session Bean maintains the state of the bean for the complete lifecycle of the application.
- Singleton Session Beans are similar to Stateless Session Beans but only one instance of the Singleton Session Bean is created in the whole application and does not terminates until the application is shut down.
- The single instance of the bean is shared between multiple clients and can be concurrently accessed.

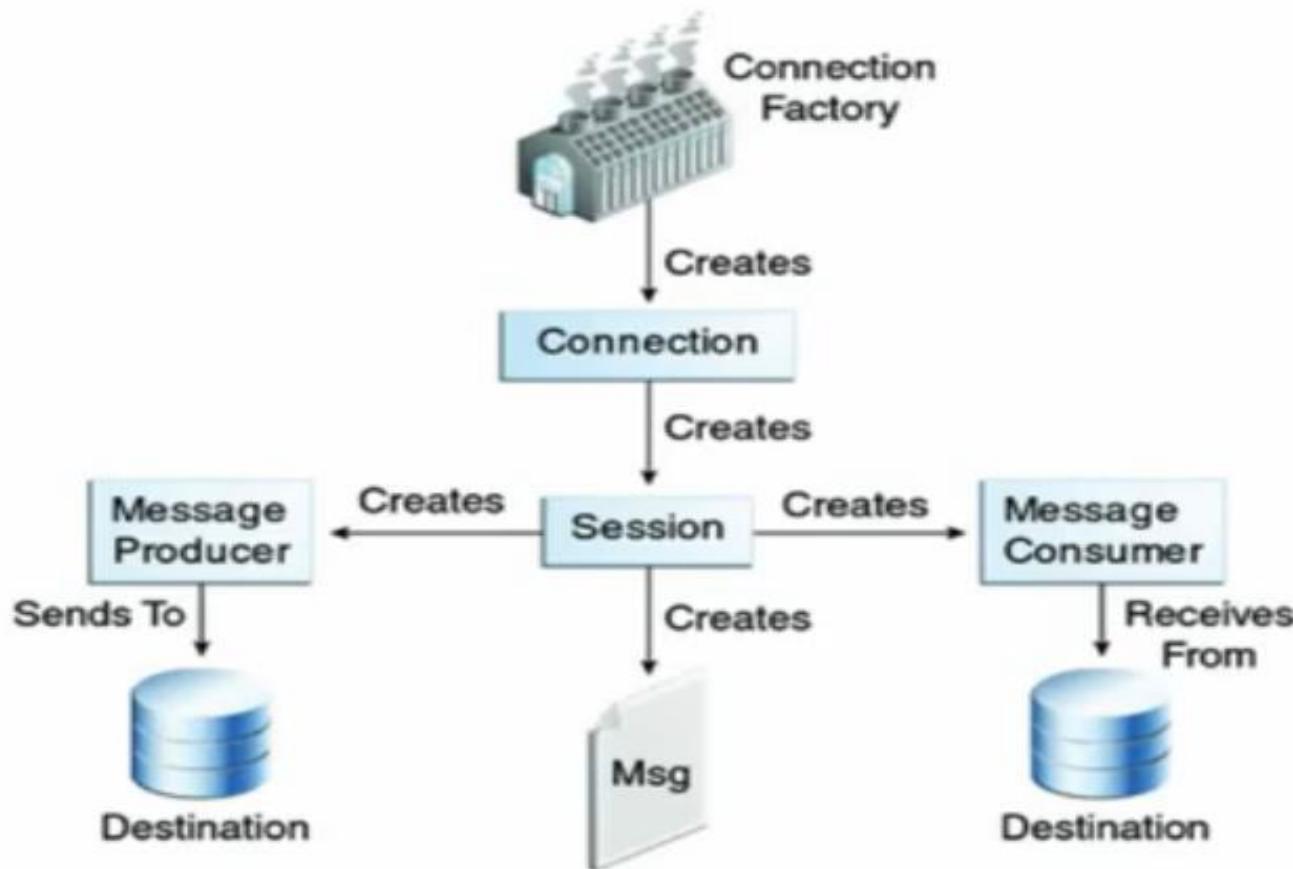
Message-Driven Bean

- An enterprise bean that allows Java EE applications to process messages asynchronously.
- Normally acts as a JMS message listener, which is similar to an event listener but receives JMS messages instead of events.
- Messages can be sent by any Java EE component (an application client, another enterprise bean, or a web component) or by a JMS application or system that does not use Java EE technology.

JMS(Java Messaging Service)

- An API that provides the facility to create, send and read messages. It provides loosely coupled, reliable and asynchronous communication.
- Advantages:
 - **Asynchronous:** To receive the message, client is not required to send request. Message will arrive automatically to the client.
 - **Reliable:** It provides assurance that message is delivered.
- Using JMS Applications communicate with each other using standard based messaging.

JMS Programming Model



More about JMS...

- Connection Factory: an object a client uses to create a connection to a provider. A connection factory encapsulates a set of connection-related parameters.
- JMS Destination : an object a client uses to specify the target of the messages. The two types of destinations in JMS are queues and topics.
- JMS connection : encapsulates a virtual connection with a JMS provider.
- JMS Sessions : a single threaded context for producing and consuming messages. Sessions are used to create – message producers, consumers, messages, queue browsers and temporary queues and topics.

More about JMS...

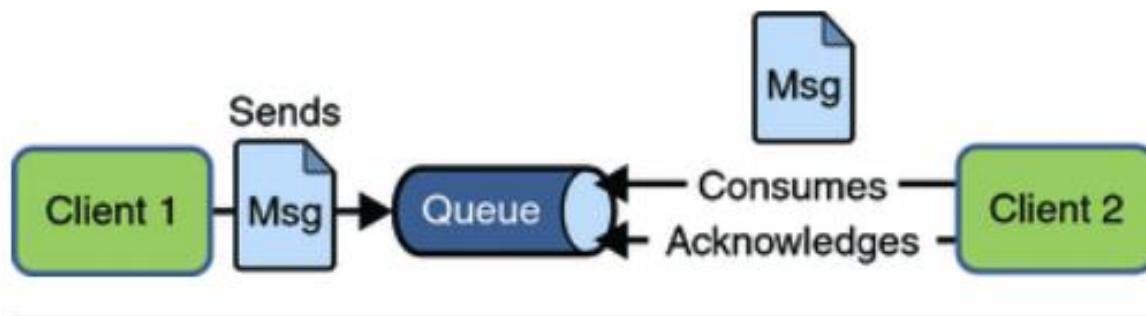
- JMS Message Listeners: an object that acts as an asynchronous event handler for messages. Implements the ‘MessageListner’ interface which contains the onMessage Method – which contains the logic that needs to be executed when a message arrives.

Messaging Domains

- Point-to-Point Messaging Domain
 - Queue
- Publisher/Subscriber Messaging Domain
 - Topic

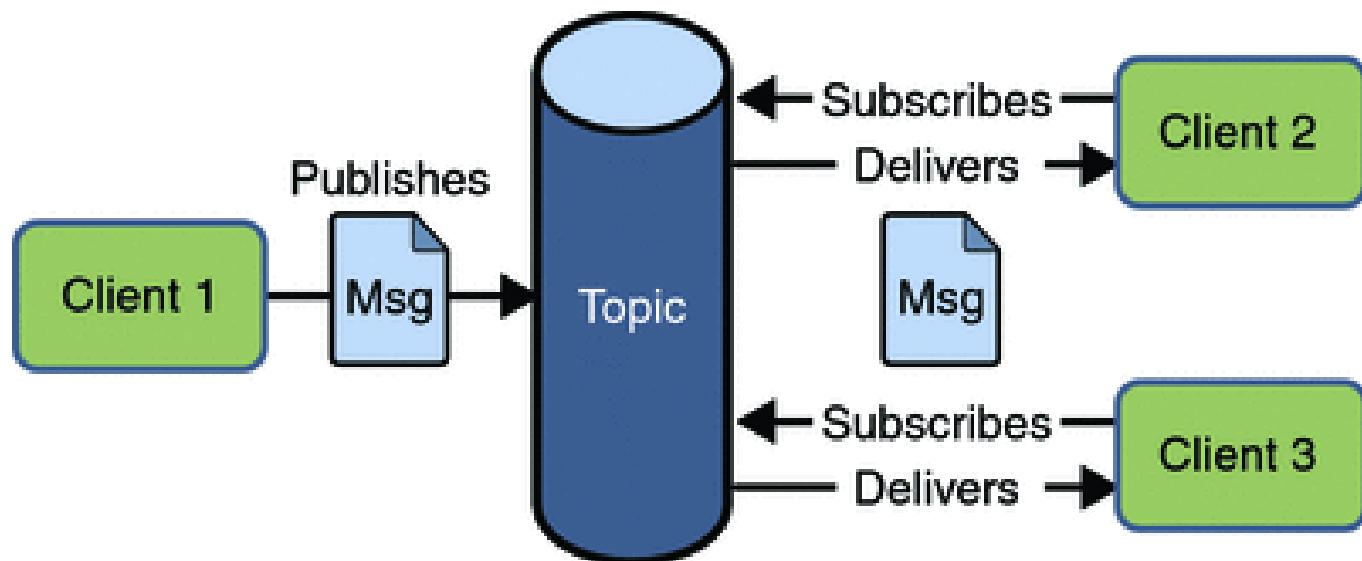
Point-to-Point (PTP) Messaging Domain

- In Ptp Model, One Message Is **Delivered To One Receiver Only**. Here, **Queue** Is Used As A Message Oriented Middleware (Mom).
- The Queue Is Responsible To Hold The Message Until Receiver Is Ready.
- In Ptp Model, There Is **No Timing Dependency** Between Sender And Receiver.



Publisher/Subscriber (Pub/Sub) Messaging Domain

- In Pub/Sub model, one message is **delivered to all the subscribers**. It is like broadcasting. Here, **Topic** is used as a message oriented middleware that is responsible to hold and deliver messages.
- In PTP model, there is **timing dependency** between publisher and subscriber.



REST Architecture

- **REpresentational State Transfer**
- Software architectural style that defines the set of rules to be used for creating web services.
- Web services that conform to the REST architectural style, are called *RESTful* Web services,
- Uses HTTP protocol
- Allows variety of data formats
- RESTful web services are light weight, highly scalable and maintainable

RESTful Web Services - Java /Jakarta(JAX-RS)

- Java EE based set of API's for implementation of Restful web services
- JAX-RS uses annotations available from Java SE 5 to simplify the development creation and deployment of Restful Web Services

JAX-RS annotations

- **@Path**
 - Relative path of the resource class/method.
- **@GET**
 - HTTP Get request, used to fetch resource.
- **@PUT**
 - HTTP PUT request, used to update resource.
- **@POST**
 - HTTP POST request, used to create a new resource.

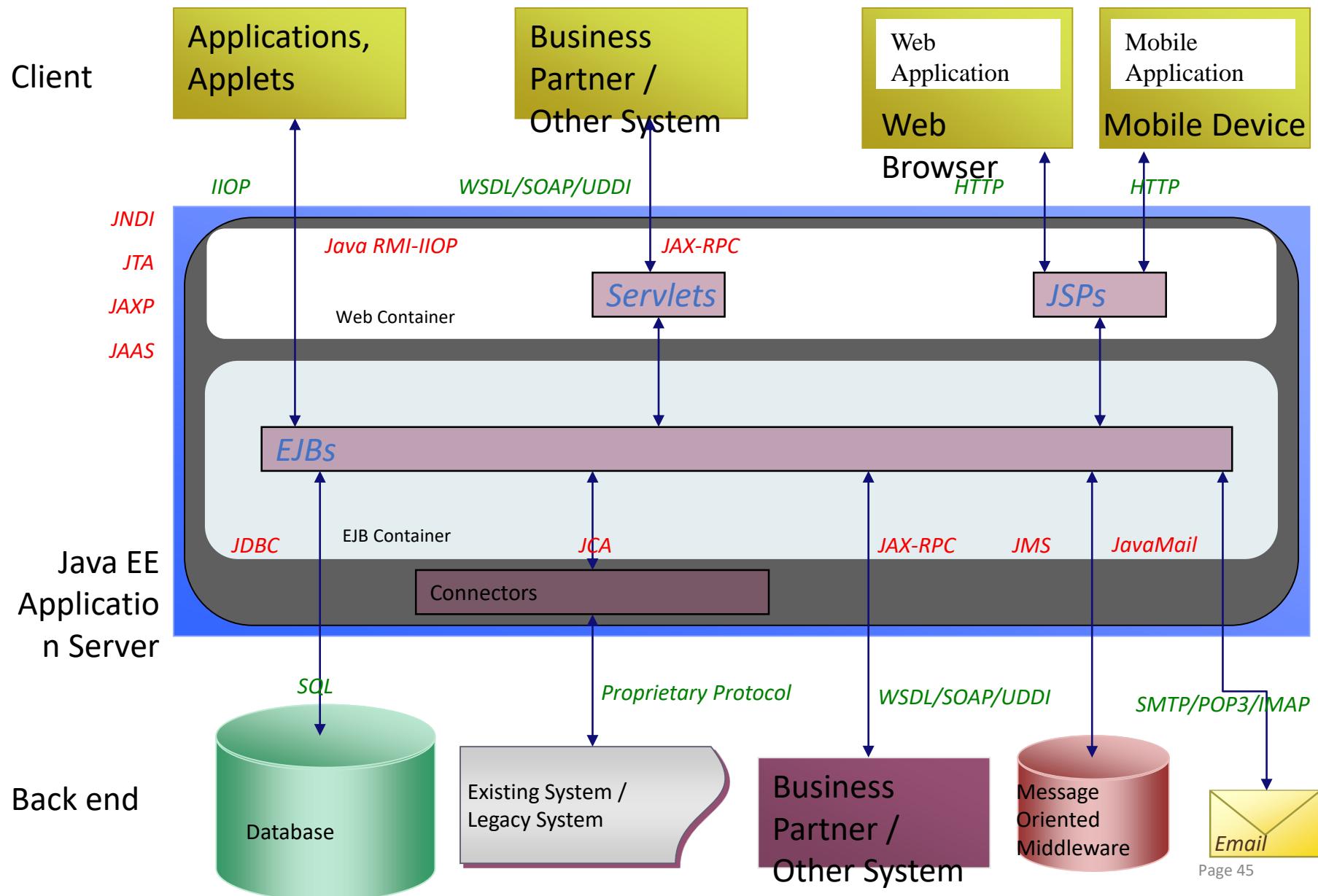
JAX-RS annotations

- **@Produces**
 - States the HTTP Response generated by web service. For example, APPLICATION/XML, TEXT/HTML, APPLICATION/JSON etc.
- **@Consumes**
 - States the HTTP Request type. For example, application/x-www-form-urlencoded to accept form data in HTTP body during POST request.

JAX-RS annotations

- **@PathParam**
 - Binds the parameter passed to the method to a value in path.
- **@QueryParam**
 - Binds the parameter passed to method to a query parameter in the path.
- **@FormParam**
 - Binds the parameter passed to the method to a form value.
- **@DefaultValue**
 - Assigns a default value to a parameter passed to the method.

Java EE Application Model + *APIs*



JNDI

- Java Naming and Directory Interface
- Utilized by JEE applications to locate resources and objects in portable fashion
 - Applications use symbolic names to find object references to resources via JNDI
 - The symbolic names and object references have to be configured by system administrator when the application is deployed.

What is a Servlet?

- Java™ objects which extend the functionality of a HTTP server
- Dynamic contents generation
- Better alternative to CGI, NSAPI, ISAPI, etc.
 - Efficient
 - Platform and server independent
 - Session management
 - Java-based

What is JSP Technology?

- Enables **separation of business logic from presentation**
 - Presentation is in the form of HTML or XML/XSLT
 - Business logic is implemented as **Java Beans or custom tags**
 - Better maintainability, reusability
- Extensible via custom tags
- Builds on Servlet technology

JEE API References

- **RMI-IIOP** (read as "RMI over IIOP") denotes the ***Java remote method invocation*** (RMI) interface over the ***Internet Inter-Orb Protocol*** (IIOP), which delivers ***Common Object Request Broker Architecture*** (CORBA) distributed computing capabilities to the Java 2 platform. Java RMI-IIOP was developed by Sun Microsystems and IBM, combining the best features of Java RMI technology with the best features of CORBA technology.
- The **Java Naming and Directory Interface (JNDI)** is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name. Like all Java APIs that interface with host systems, JNDI is independent of the underlying implementation. Additionally, it specifies a service provider interface (SPI) that allows directory service implementations to be plugged into the framework. The implementations may make use of a server, a flat file, or a database; the choice is up to the vendor.
- **JavaServer Pages (JSP)** is a Java technology that allows software developers to create dynamically generated web pages, with HTML, XML, or other document types, in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.
- **Servlets** are Java programming language objects that dynamically process requests and construct responses. The **Java Servlet API** allows a software developer to add dynamic content to a Web server using the Java platform.

JEE API References...

- **Java API for XML-based RPC** (JAX-RPC) allows a Java application to invoke a Java-based Web Service with a known description while still being consistent with its WSDL description. It can be seen as Java RMIs over Web services. JAX-RPC 2.0 was renamed to JAX-WS 2.0 (Java API for XML Web Services).
- **JDBC (Java Database Connectivity)** is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.
- The **Java Transaction API (JTA)** is one of the Java EE APIs allowing distributed transactions to be done across multiple XA resources.
- The **Java API for XML Processing** or **JAXP** (pronounced jaks-p), is one of the Java XML programming APIs. It provides the capability of validating and parsing XML documents.
- **Java Authentication and Authorization Service**, or **JAAS**, pronounced "Jazz", is a Java security framework for user-centric security to augment the Java code-based security.

JEE API References...

- **Java EE Connector Architecture (JCA)** is a Java-based technology solution for connecting application servers and enterprise information systems (EIS) as part of enterprise application integration (EAI) solutions. While JDBC is specifically used to connect Java EE applications to databases, JCA is a more generic architecture for connection to legacy systems (including databases).
- The **Java Message Service (JMS)** API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients.
- **JavaMail** is a Java API used to receive and send email via SMTP,POP3 and IMAP.

Questions?

- Feel free to drop a mail to sachit.talagala@ifsworld.com
- Questions like
 - Paper structure
 - Weight of marks for mid semester and final exam

Should be directed to your course co-Ordinator 😊



Cloud Computing

Enterprise Application Development

INTRODUCTION TO CLOUD COMPUTING

IFS ACADEMY

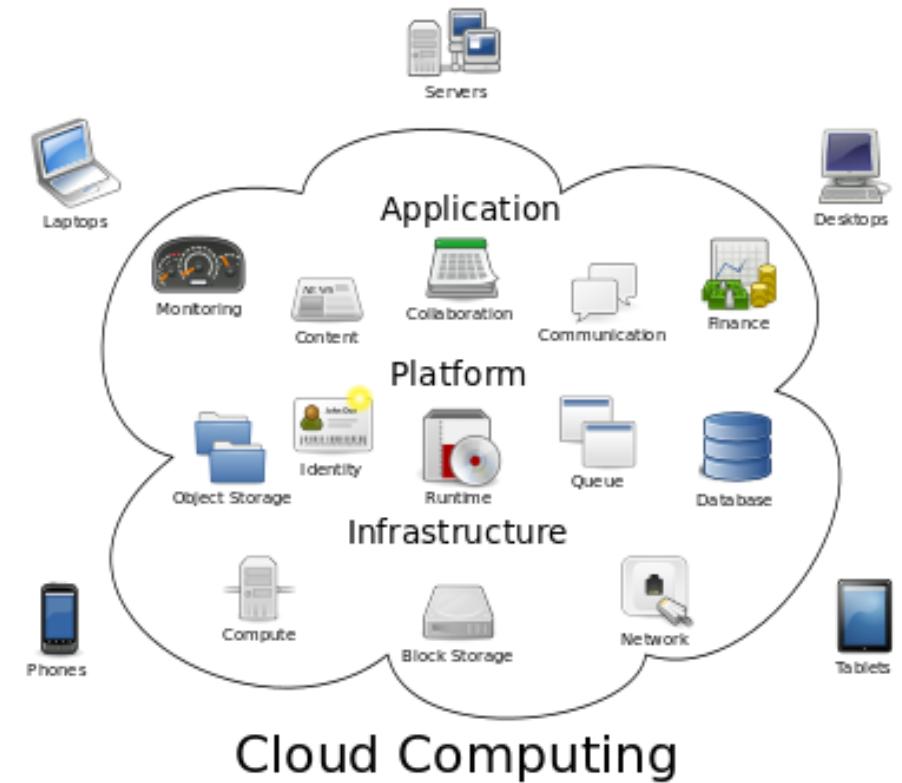
What is Cloud Computing ?

Agenda

- | | |
|--|---|
| <p>01 Introduction</p> <p>02 Main Concepts</p> <p>03 The History</p> <p>04 Pros and Cons</p> | <p>05 Associated Technologies</p> <p>06 Architecture</p> <p>07 Cloud Models</p> <p>08 Cloud Development</p> |
|--|---|

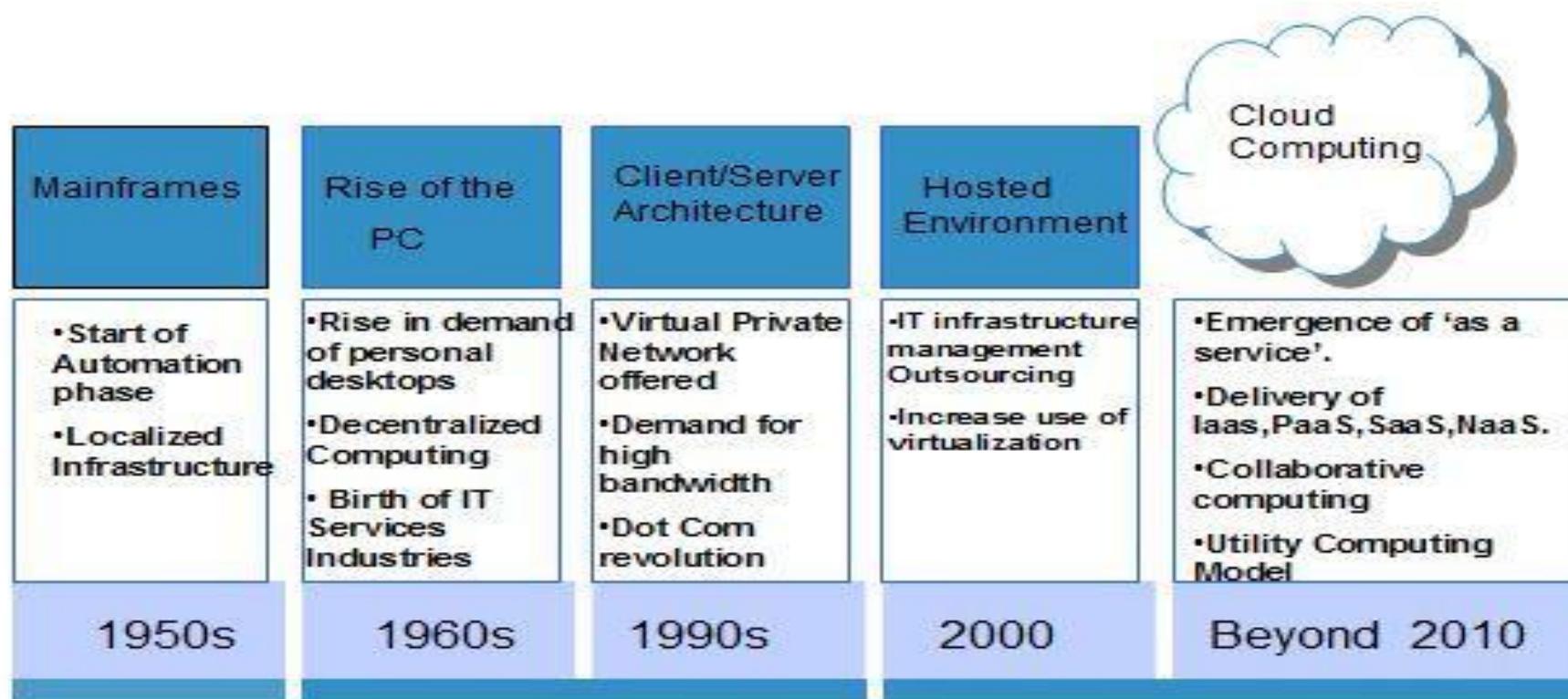
What is Cloud Computing ?

- Cloud computing is the delivery of computing as a service rather than a product.
- Shared resources, software and information are provided to computers and other devices as a utility over a network.
 - Cloud computing metaphor – For a user, the network elements representing the provider-rendered services are invisible, as if obscured by a cloud.



History of Cloud Computing

- The concept of Cloud Computing came into existence in 1950 with implementation of mainframe computers, accessible via thin/static clients.
- Since then, cloud computing has evolved from static clients to dynamic ones, from software to services



Main Concepts

1. Deployment Models

Deployment models define the type of access to the cloud.

Eg: Where the cloud is located.

Cloud can have any of the four types of access: Public, Private, Hybrid and Community.

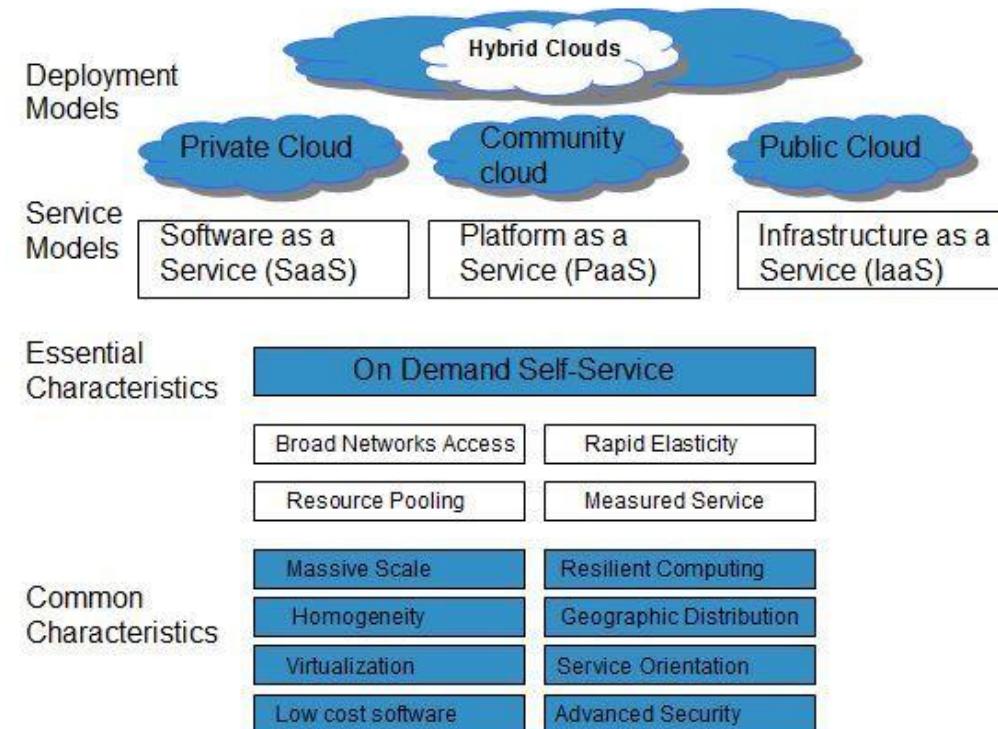
2. Service Models

Service Models are the reference models on which the Cloud Computing is based.

These can be categorized into three basic service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

Characteristics of Cloud Computing

On Demand Self Service - Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them.



Characteristics of Cloud Computing

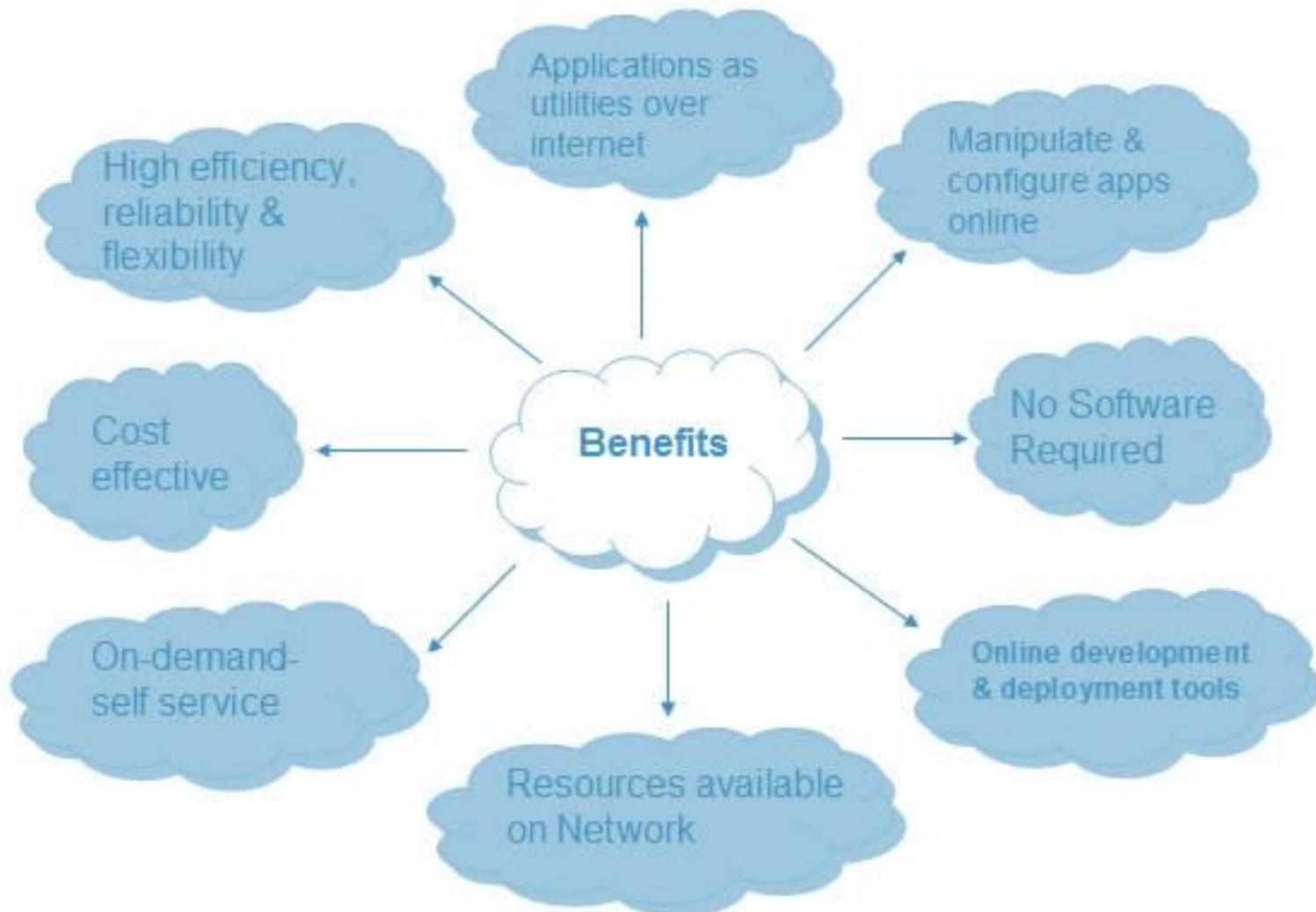
Broad Network Access - Since Cloud Computing is completely web based, it can be accessed from anywhere and at any time.

Resource Pooling - Cloud Computing allows multiple tenants to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.

Rapid Elasticity - It is very easy to scale up or down the resources at any time.

Measured Service - It is possible to monitor the resources which were used by the customers and currently assigned to customers.

Benefits of Cloud Computing



What is Cloud Computing?



Risks of Cloud Computing

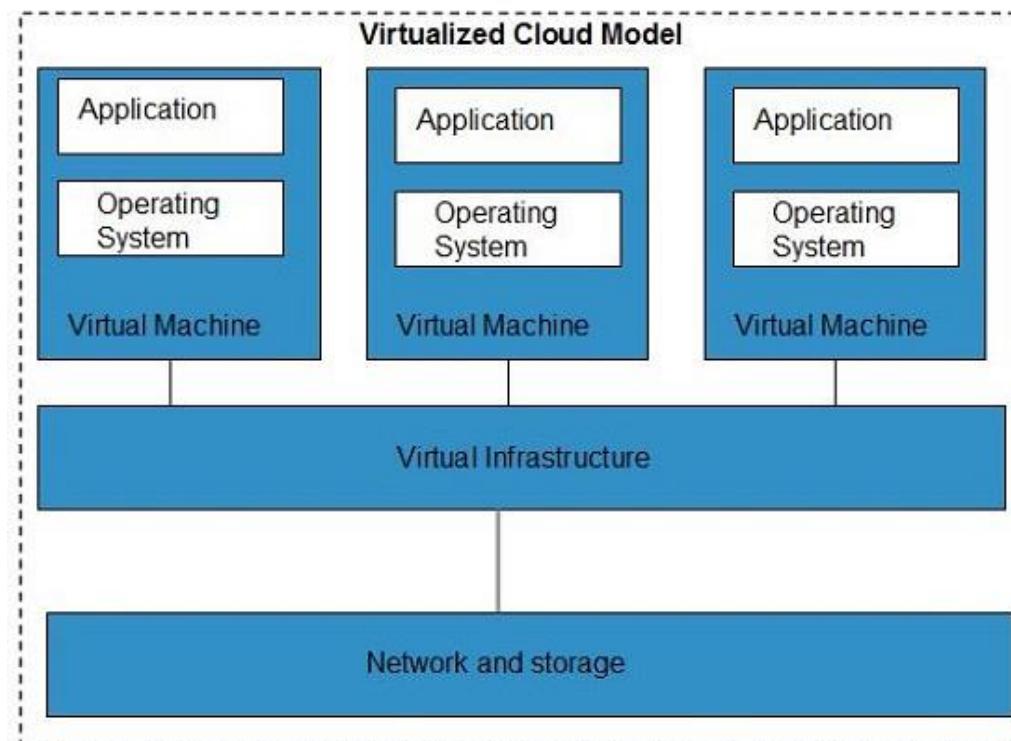
1. **Security & Privacy** - Since data management and infrastructure management in cloud is provided by third-party, it is always a risk to handover the sensitive information to such providers.
2. **Lock-In** - It is very difficult for the customers to switch from one Cloud Service Provider (CSP) to another. It results in dependency on a particular CSP for service.
3. **Isolation Failure** - This risk involves the failure of isolation mechanism that separates storage, memory, routing between the different tenants.
4. **Management Interface Compromise** - In case of public cloud provider, the customer management interfaces are accessible through the Internet.
5. **Insecure or Incomplete Data Deletion** - It is possible that the data requested for deletion may not get deleted.

Cloud Computing Technologies

- There are certain technologies that are working behind the cloud computing platforms making cloud computing flexible, reliable, usable. These technologies are listed below:
 - Virtualization
 - Service-Oriented Architecture (SOA)
 - Grid Computing
 - Utility Computing

Virtualization

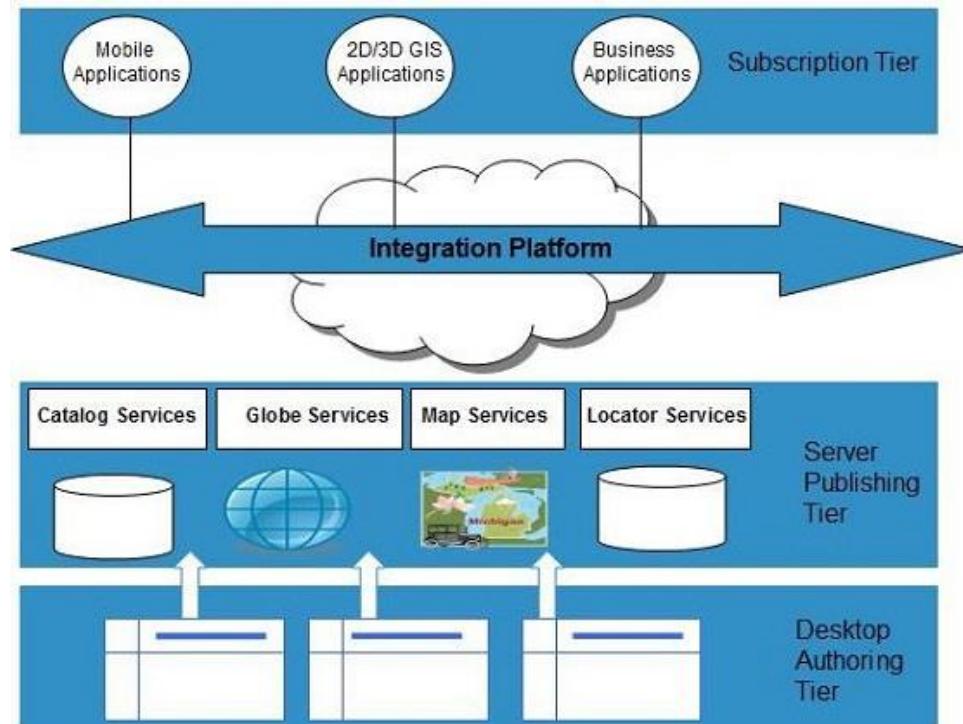
- Virtualization is a technique which allows to share single physical instance of an application or resource among multiple organizations or tenants (customers).
- It offers virtual isolation among multiple tenants and therefore, the organizations can use and customize the application as though they each has its own instance running.



Service Oriented Architecture (SOA)

SOA allows us to use an application as a service for the other applications regardless the type of vendor, product or technology.

- Therefore, it is possible to exchange data between applications of different vendors without additional programming.

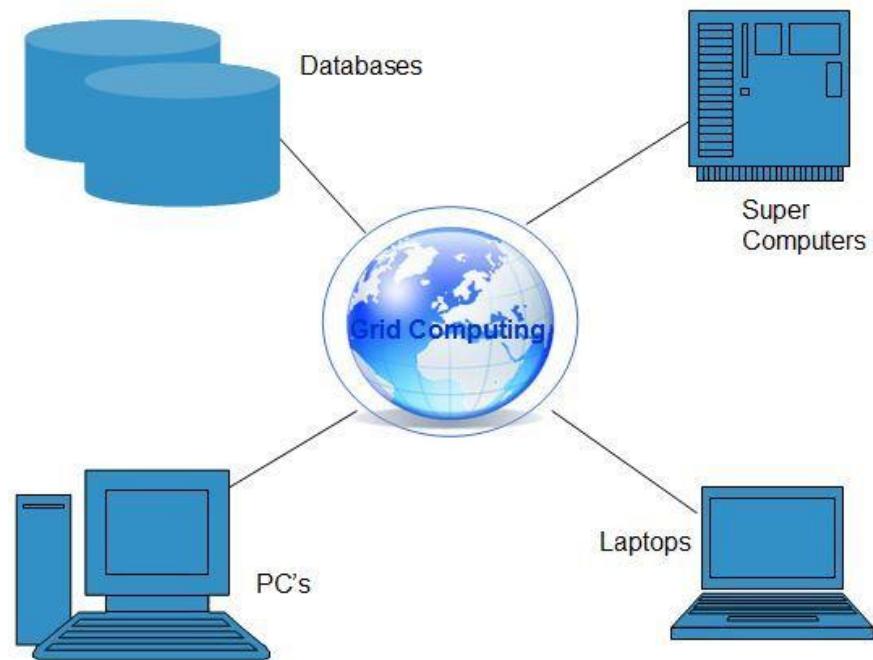


Grid Computing

Grid Computing refers to distributed computing in which a group of computers from multiple locations are connected with each other to achieve common objective.

These computer resources are heterogeneous and geographically dispersed.

- Grid Computing breaks complex task into smaller pieces.
- These smaller pieces are distributed to CPUs that reside within the grid.



Utility Computing

- Utility computing is based on Pay per Use model.
- It offers computational resources on demand as a metered service.
- Cloud computing, grid computing, and managed IT services are based on the concept of Utility computing

Cloud Computing Architecture

We can broadly divide the cloud architecture into two parts.

1. Front end

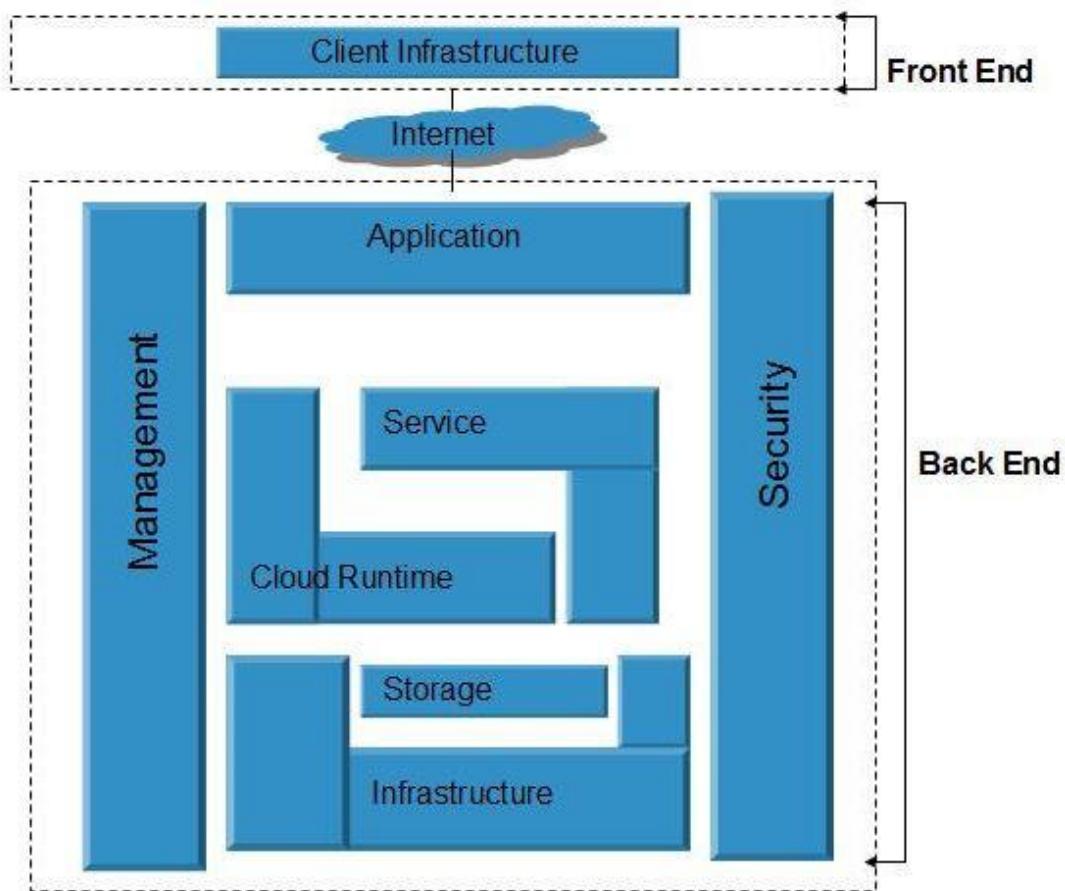
- Refers to the client part of cloud computing system.
- Consists of interfaces and applications that are required to access the cloud computing platforms.
- Eg: web browser.

2. Back end

- Refers to the cloud itself.
- Consists of all the resources required to provide cloud computing services.
- Eg: data storage, virtual machines, security mechanisms, services, deployment models, servers, etc.

Cloud Computing Architecture

- Front end and the back end are connected through a network (usually Internet).



Cloud Computing Infrastructure

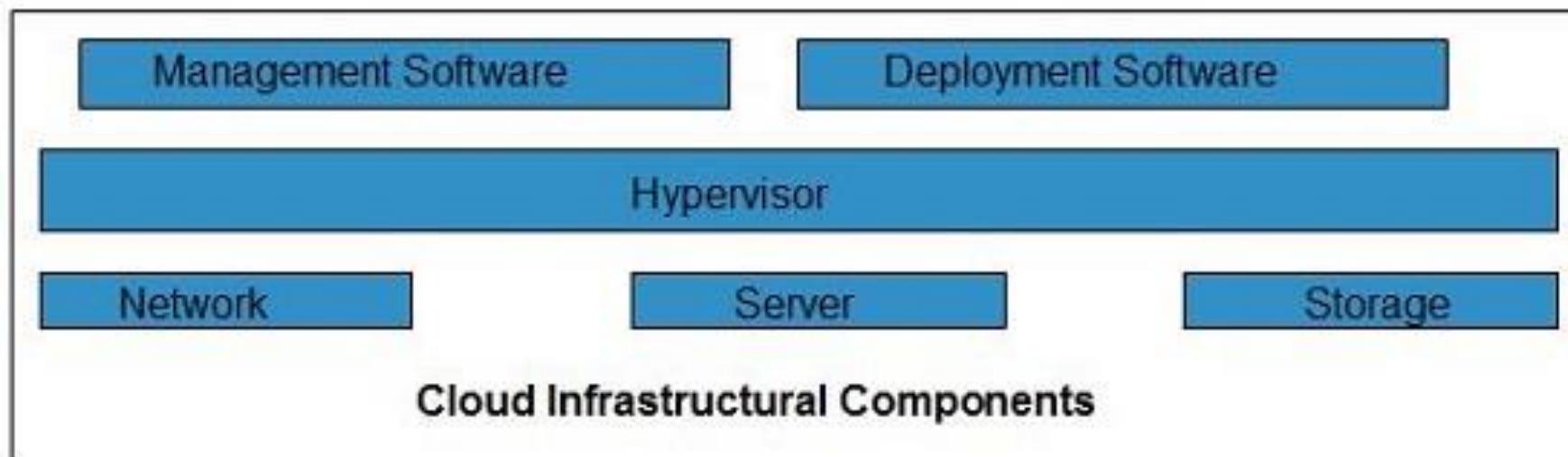
1. Hypervisor

A firmware or low-level program that acts as a Virtual Machine Manager.

It allows to share the single physical instance of cloud resources between several tenants.

2. Management Software

Helps to maintain and configure the infrastructure.



Cloud Computing Infrastructure

3. Deployment Software

Helps to deploy and integrate the application on the cloud.

4. Network

The key component of cloud infrastructure.

Allows to connect cloud services over the Internet.

5. Server

Helps to compute the resource sharing and offer other services such as resource allocation and deallocation, monitoring resources, security, etc.

6. Storage

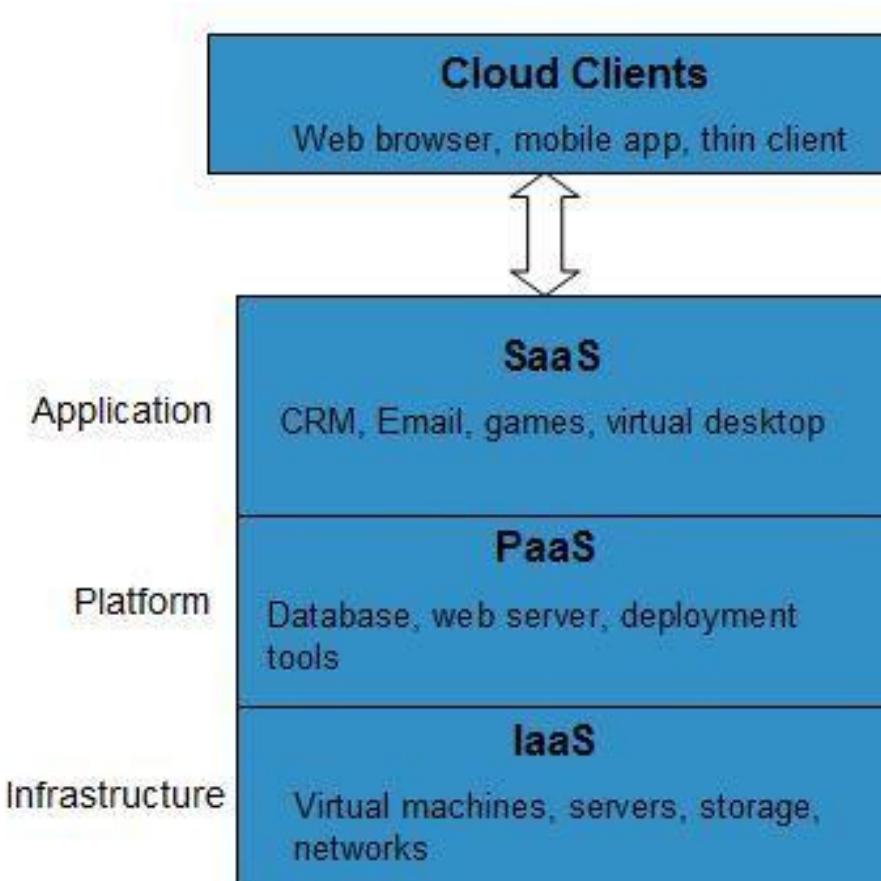
Cloud uses distributed file system for storage purpose.

If one of the storage resource fails, then it can be extracted from another one which makes cloud computing more reliable

Service Models

Service Models

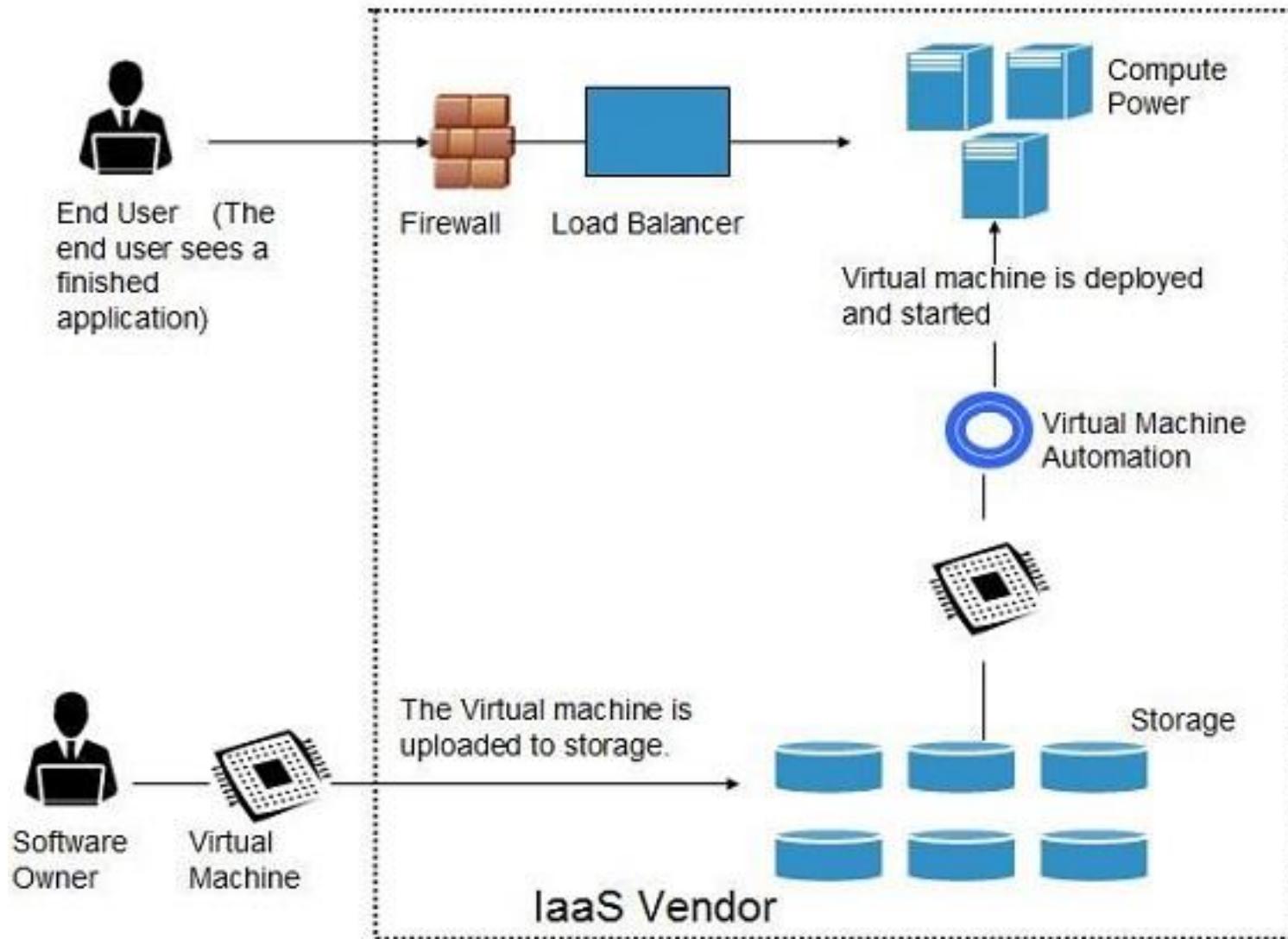
- Each of the service models make use of the underlying service model.



Infrastructure as a Service (IaaS)

- IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.
- Apart from these resources, the IaaS also offers:
 - Virtual machine disk storage
 - Virtual local area network (VLANs)
 - Load balancers
 - IP addresses
- All of the above resources are made available to end user via server virtualization. Moreover, these resources are accessed by the customers as if they own them.

Infrastructure as a Service (IaaS)



Infrastructure as a Service

Benefits

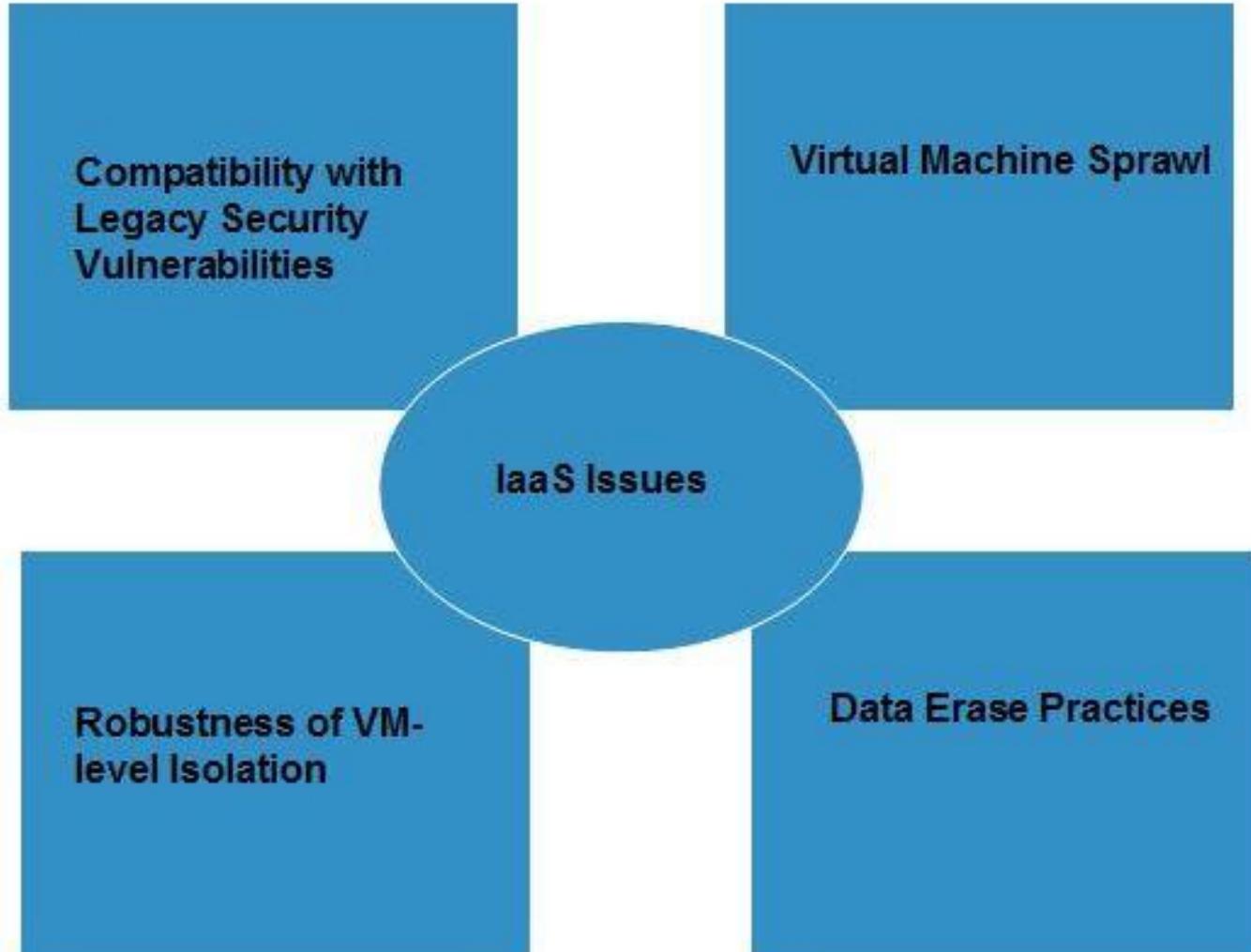
Full Control of the computing resources through Administrative Access to VMs

Flexible and Efficient renting of Computer Hardware

Portability, Interoperability with Legacy Applications

Infrastructure as a Service

Issues



Infrastructure as a Service

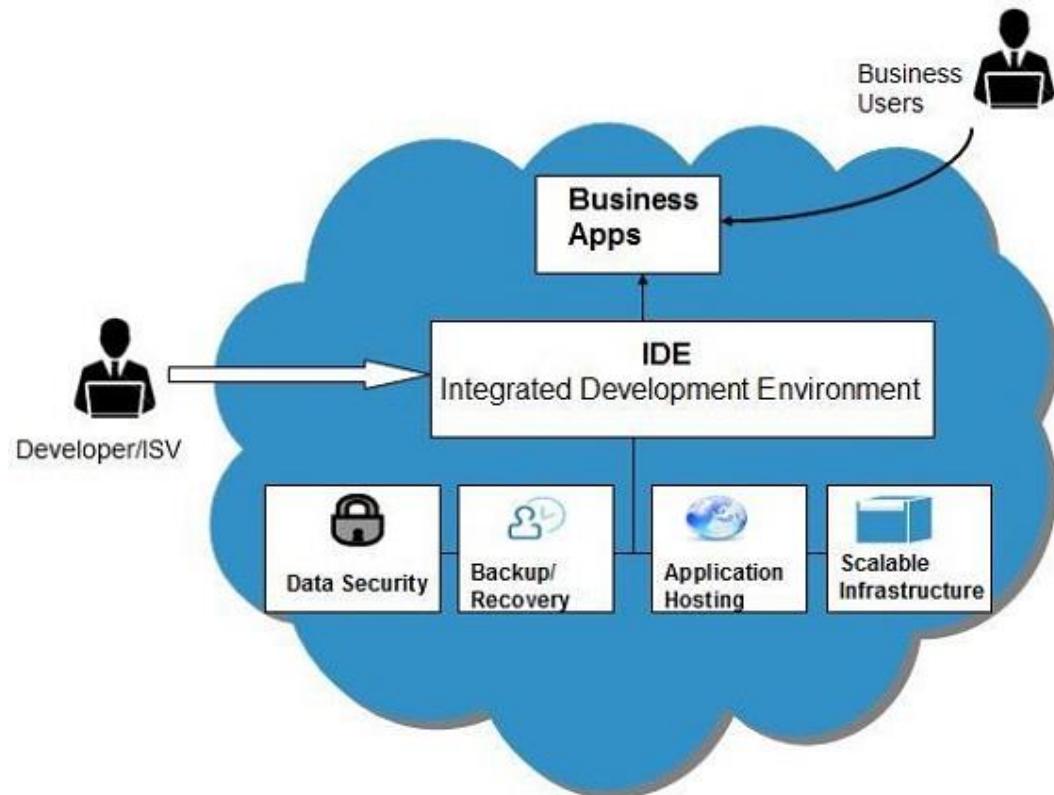
Providers



Converged Infrastructure

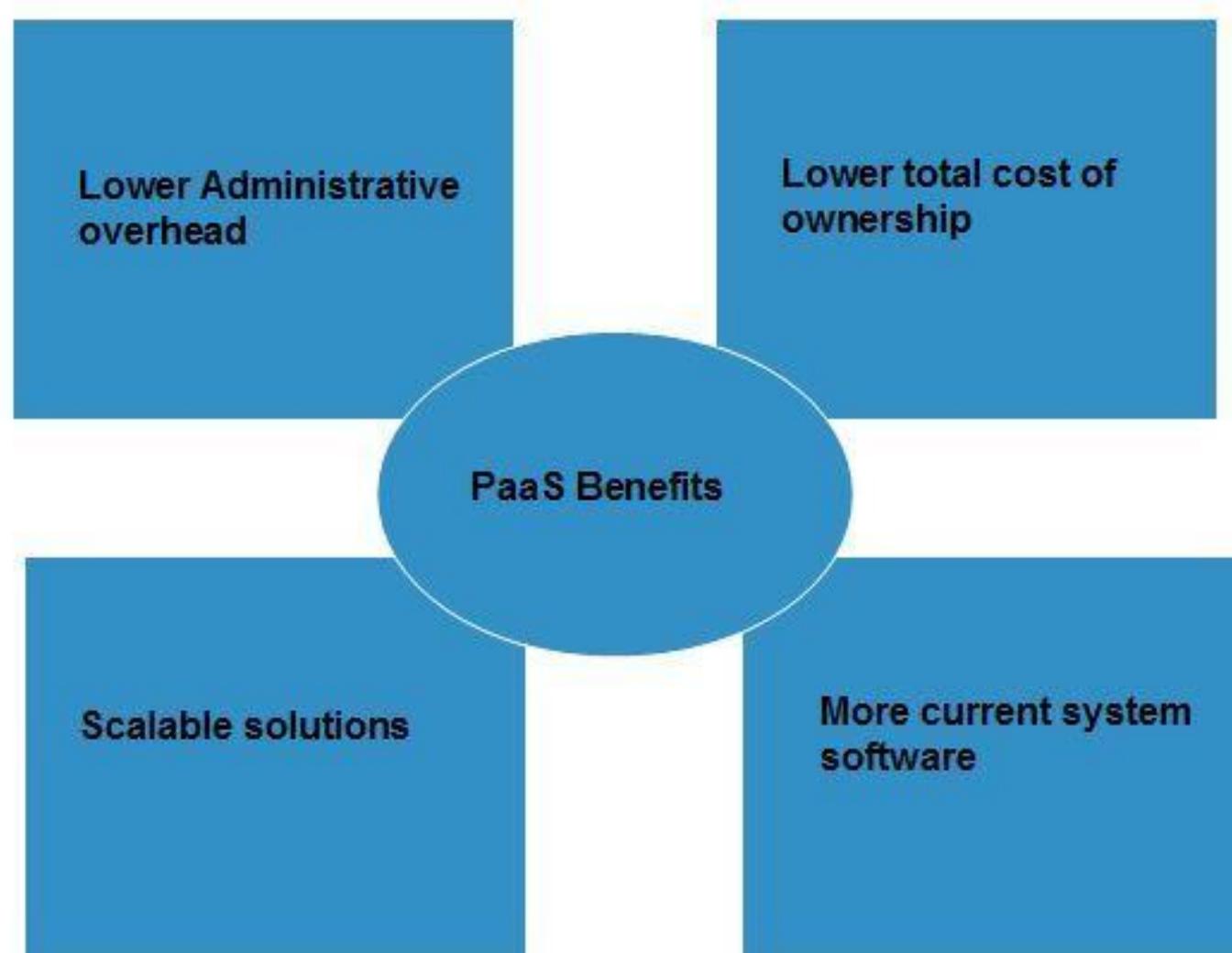
Platform as a Service (PaaS)

- PaaS offers the runtime environment for applications.
- It also offers development & deployment tools required to develop applications.



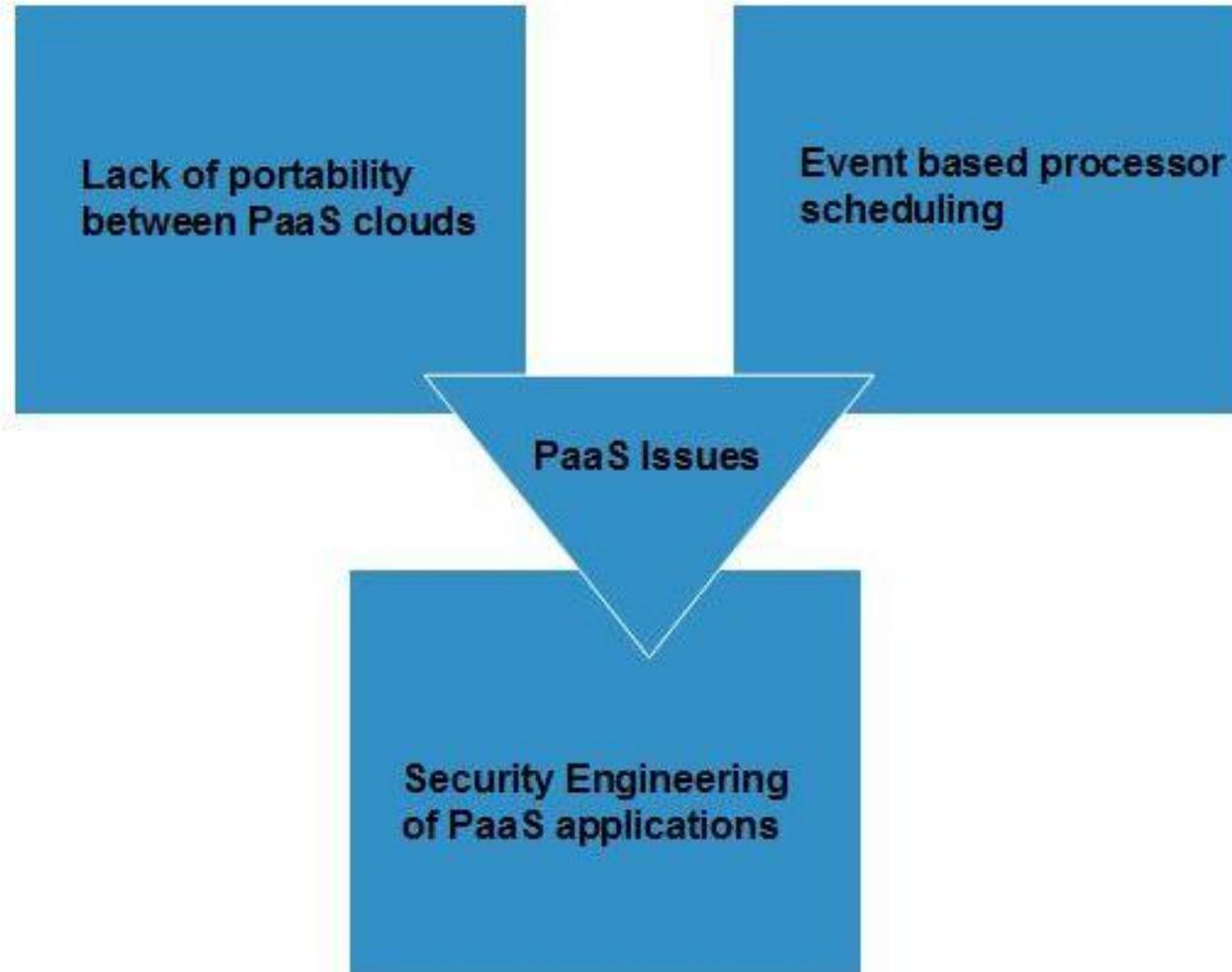
Platform as a Service

Benefits



Platform as a Service

Issues



Platform as a Service

Providers



Google App Engine



Software as a Service (SaaS)

- SaaS offers software applications as a service to the end users.
- It refers to a software that is deployed on a hosted service and is accessible via Internet.
- Application deployment requires little or no client-side software installation.
- The license to the software may be subscription based or usage based.
- All users are running same version of the software.

Software as a Service

Benefits

1. **Cost Effective** – SaaS applications do not require any maintenance at end user side.
2. **Scalability** – they can be scaled up or down depending on the demand.
3. **Ease of Use** - SaaS application deployment requires a little or no client-side software installation.
4. **Efficient use of Licenses** - the client can have single license for multiple computers running at different locations which reduces the licensing cost.
5. **Multitenant Solutions** - multiple users to share a single instance of resources in virtual isolation.

Software as a Service

Issues

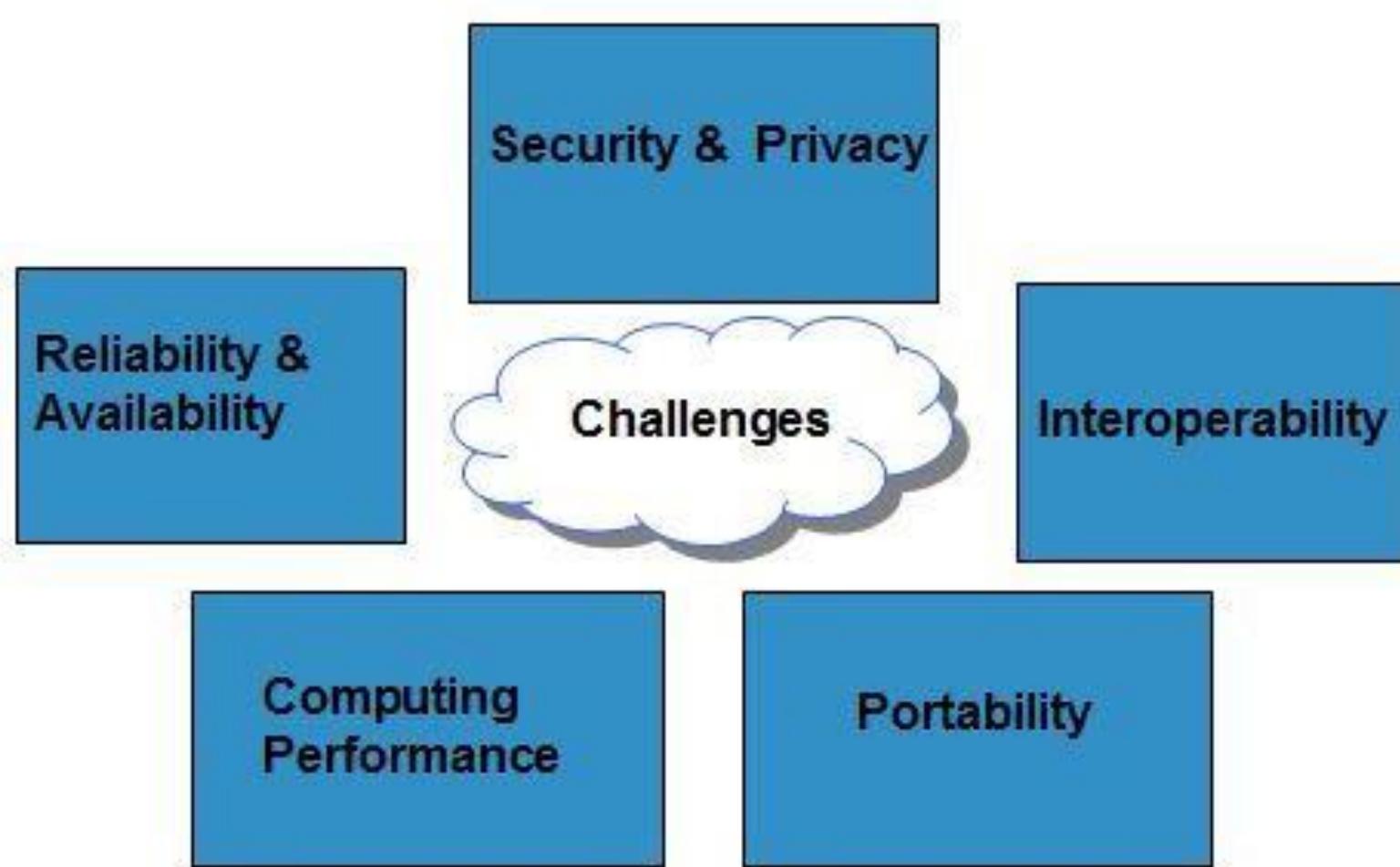
1. **Browser Based Risks** - If the consumer visits malicious website and browser becomes infected, the subsequent access to SaaS application might compromise the consumer's data.
2. **Network Dependence** - the SaaS application can be delivered only when network is continuously available. Also, the network connection should be reliable.

Platform as a Service

Providers



Cloud Computing Challenges



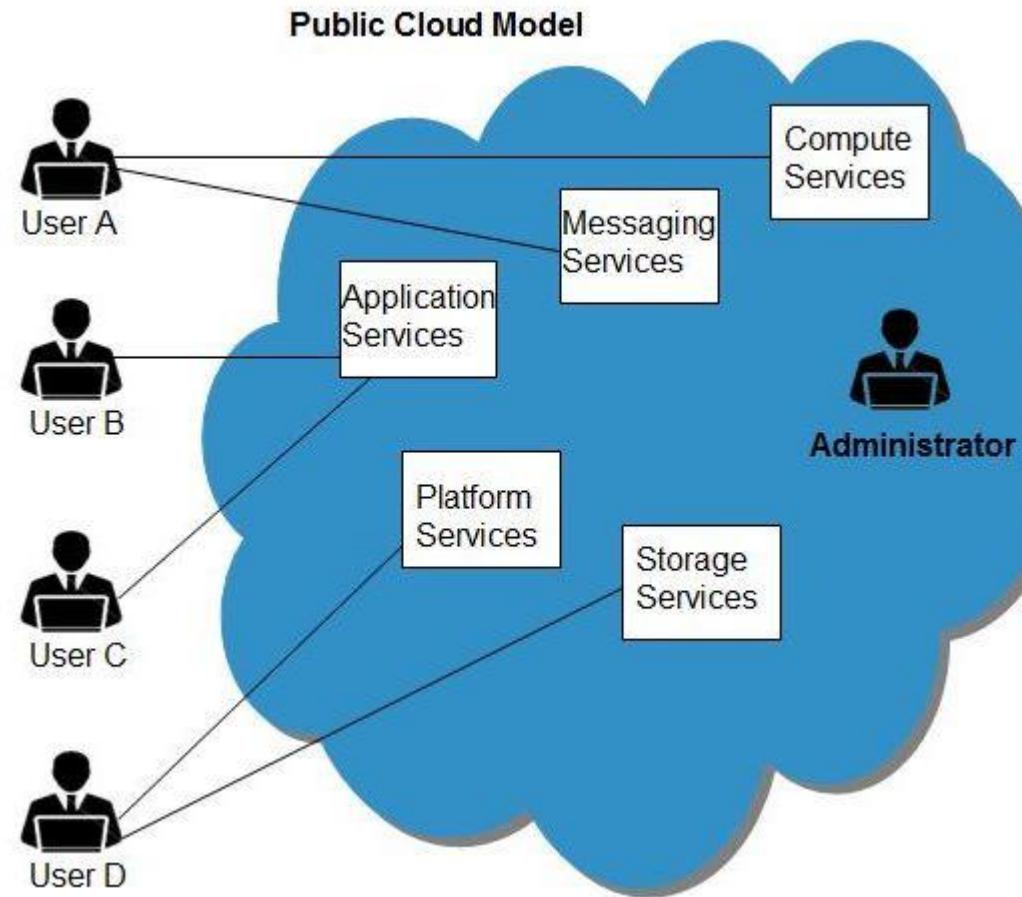


Pause II

Deployment Models

Public Cloud Model

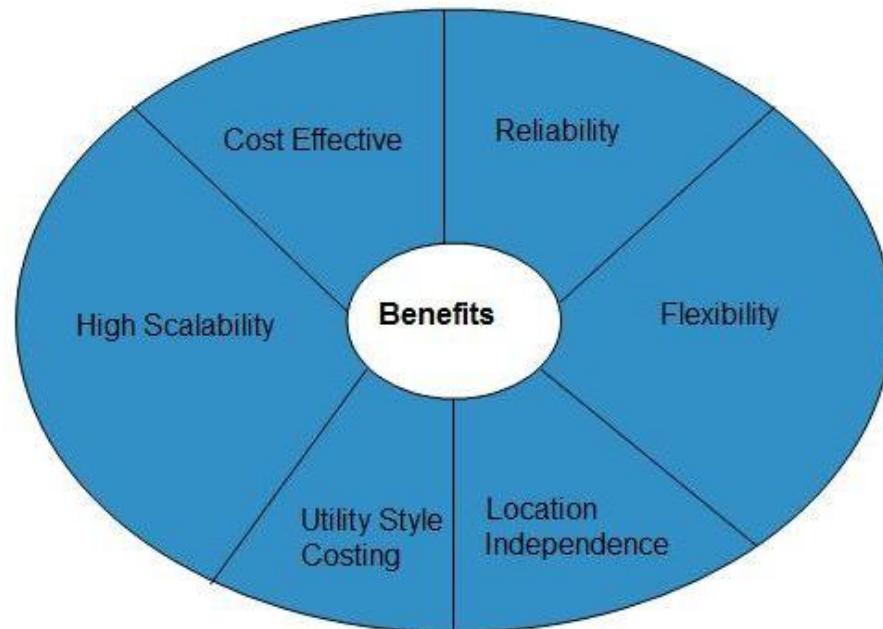
The Public Cloud allows systems and services to be easily accessible to general public.



Public Cloud Model

Benefits

1. **Cost Effective** - public cloud share the same resources with a large number of consumers.
2. **Reliability** - public cloud employs large number of resources from different locations. If any of the resources fail, the cloud can employ another one.



Public Cloud Model

Benefits

3. **Flexibility** – public cloud services can be integrated with other services such as private clouds.
4. **Location Independence** – public cloud services are delivered through the internet, thus they are accessible from anywhere in the world.
5. **Utility Style Costing** - public cloud is also based on pay-per-use model and resources are accessible whenever consumer needs it.
6. **High Scalability** - cloud resources are made available on demand from a pool of resources. They can be scaled up or down according the requirement.

Public Cloud Model

Disadvantages

1. Low Security

Data is hosted off-site, and resources are shared publicly in public cloud model.

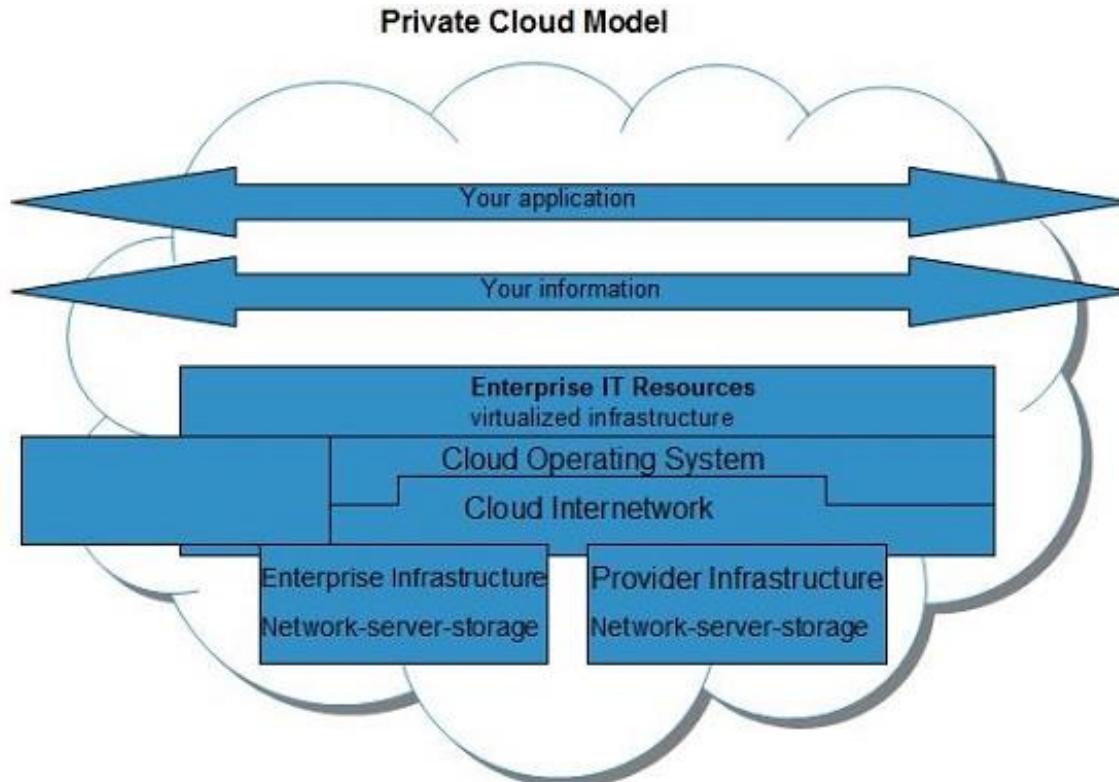
Therefore, it does not ensure higher level of security.

2. Less Customizable

Public cloud is comparatively less customizable than private cloud.

Private Cloud Model

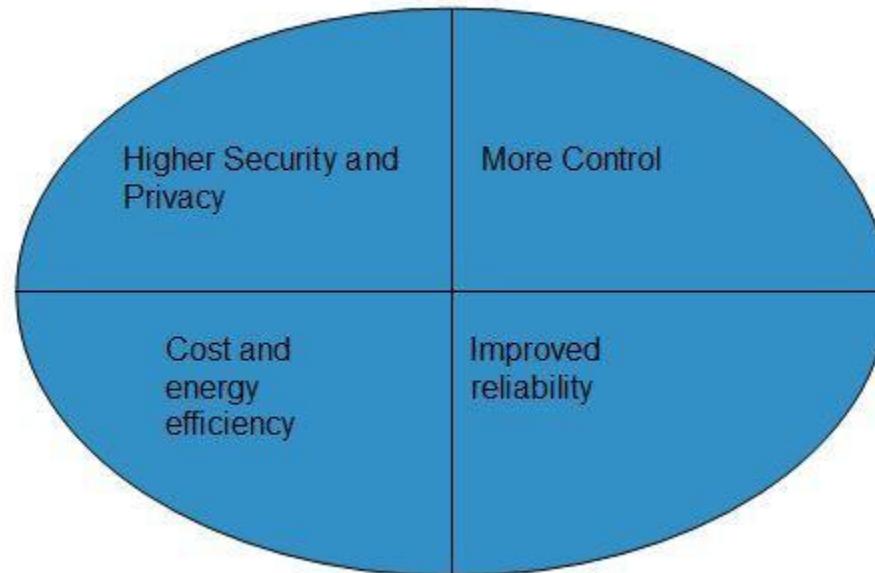
- Private Cloud allows systems and services to be accessible within an organization.
- It is operated only within a single organization. However, It may be managed internally or by third-party.



Private Cloud Model

Benefits

1. **Higher Security & Privacy** - private cloud operations are not available to general public and resources are shared from distinct pool of resources.
2. **More Control** - private clouds have more control on its resources and hardware than public cloud because it is accessed only within an organization



Private Cloud Model

Benefits

3. **Cost & Energy Efficiency** – private cloud resources are not as cost effective as public clouds but they offer more efficiency than public cloud.
4. **Improved Reliability** – private cloud employs a number of resources pooled within the cloud. If any of these resources fail, the cloud can employ another one.

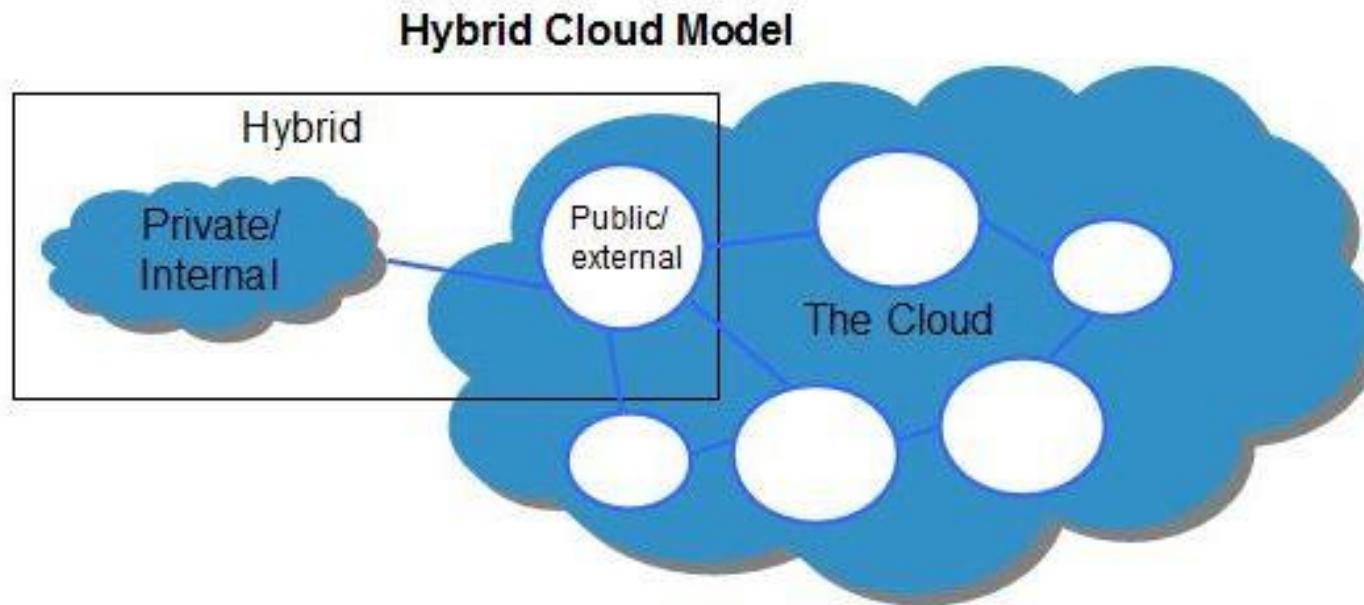
Private Cloud Model

Disadvantages

1. Restricted Area - private cloud is only accessible locally within the organization.
2. Inflexible Pricing – purchasing new hardware to cater for the increasing demand is costly.
3. Limited Scalability - private cloud can be scaled only within capacity of internal hosted resources.
4. Additional Skills – organizations require additional skills and expertise in order to maintain a private cloud deployment.

Hybrid Cloud Model

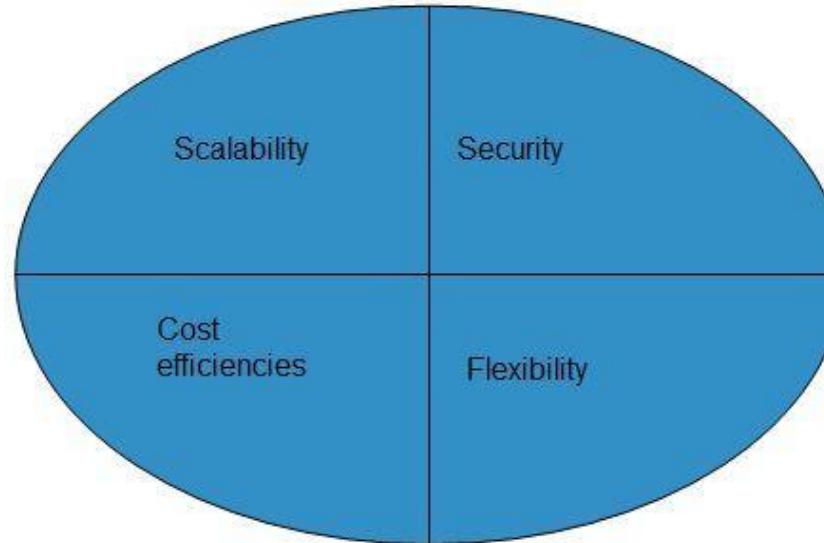
- The Hybrid Cloud is a mixture of public and private cloud.
- Non-critical activities are performed using public cloud while the critical activities are performed using private cloud.



Hybrid Cloud Model

Benefits

1. **Scalability** - It offers both features of public cloud scalability and private cloud scalability.
2. **Flexibility** - It offers both secure resources and scalable public resources.
3. **Cost Efficiencies** - public cloud are more cost effective than private, therefore hybrid cloud can have this saving.
4. **Security** - Private cloud in hybrid cloud ensures higher degree of security.



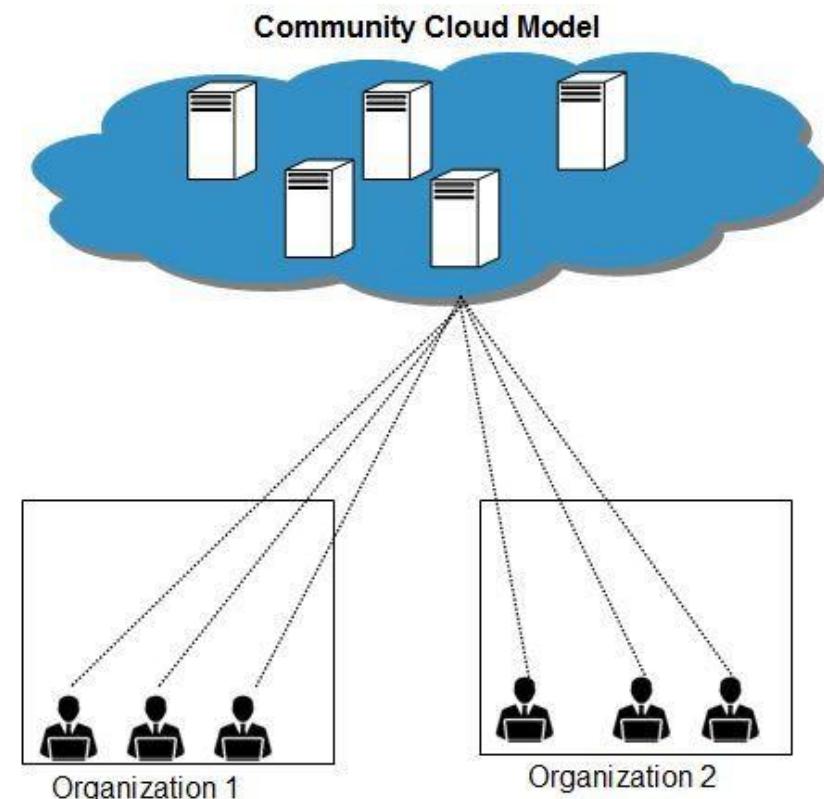
Hybrid Cloud Model

Disadvantages

1. **Networking Issues** - Networking becomes complex due to presence of private and public cloud.
2. **Security Compliance** - it is necessary to ensure that cloud services are compliant with organization's security policies.
3. **Infrastructural Dependency** - the hybrid cloud model is dependent on internal IT infrastructure; therefore, it is necessary to ensure redundancy across data centers.

Community Cloud Model

- The Community Cloud allows system and services to be accessible by group of organizations.
- It shares the infrastructure between several organizations from a specific community.
 - It may be managed internally or by a third-party.



Community Cloud Model

Benefits

1. **Cost Effective** – community cloud is more cost effective compared to the private cloud.
2. **Resource Sharing** - community cloud provides an infrastructure to share cloud resources and capabilities among several organizations.
3. **Security** - community cloud is comparatively more secure than the public cloud (but less secure than the private cloud).

Community Cloud Model

Disadvantages



Cloud Development

Cloud Development

Introduction

Cloud Development / Cloud Native Development

- Cloud development and cloud-native development, both refer to building a software application designed to run in a cloud computing environment within and specifically for that environment.
- In a traditional software development approach, software is developed and tested on a computer and then run on either physical servers or a private or cloud as a production or 'live' environment.
- In cloud development, the software is developed, tested and run within a cloud environment.

Cloud Development

Why Develop In-Cloud?

- Cloud development or in-cloud development are two different ways to say the same thing. The latter is less common but removes any ambiguity between developing software for the cloud and developing a cloud infrastructure, which is of course a very different thing.
- Whatever the preferred terminology, the core point is that if an app is intended to be run in a cloud environment, it is generally far more efficient to also develop and test it in the same cloud environment.
- Over the past couple of years there have been several developments leading to the belated mainstreaming of cloud development, which has been around for almost 10 years without, until recently, establishing itself as the software development standard:

Cloud Development

Why Develop In-Cloud?

The Benefits Of Cloud Development

- Cost-cutting and investment
- Flexibility
- Collaboration Efficiency
- Scalability
- Document Management

Cloud Development

Why Develop In-Cloud?

Examples of successful cloud development projects

Salesforce

Salesforce offers a SaaS platform in the cloud. It is a solution for both business-to-business and business-to-customer transactions.

Slack

Slack is a cloud service headquartered in the United States that uses tools and services to facilitate internal team collaboration.

Nerds Support Cloud

Nerd Support Cloud Computing Miami Services includes a number of different services in its cloud system. The Nerd Support cloud supports medium and small-sized enterprises with all IT-related services. They consist of SaaS platforms, hardware upgrades and maintenance, and security. They allow them to function at peak performance.

Google Cloud

Google Cloud Platform is a cloud service provided by Google. It is built on Google's same infrastructure for its end-user products, such as Google and YouTube.

Adobe Creative Cloud

Adobe Creative Cloud is a collection of Adobe Systems products and services. It provides members with access to  software for graphic design, video editing, web design, photography, and others.

Cloud Development

Cloud Native

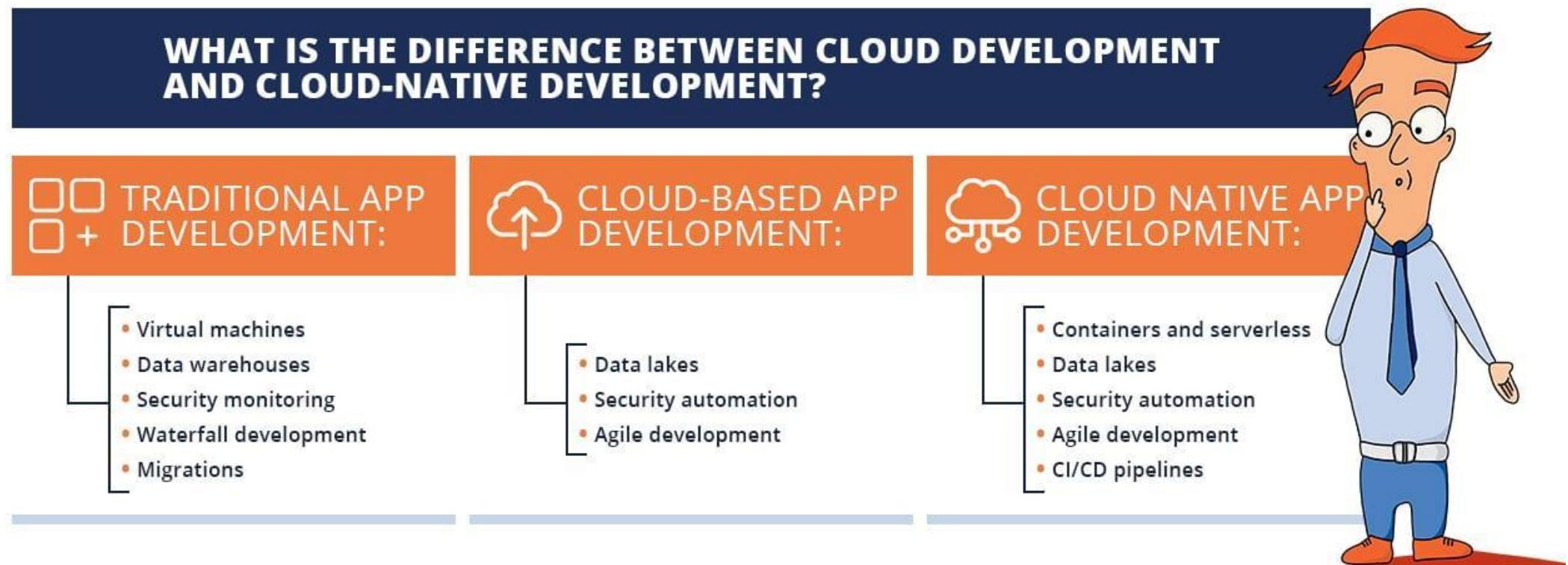
- Cloud development or cloud-based development and cloud-native development are often used interchangeably. There is, however, key distinctions between the two. Cloud development simply means writing code in the cloud, or on a local machine directly connected to the cloud environment, to where it is transferred for testing.
- Cloud development requires only a browser or online interface that is connected to a cloud-based infrastructure.
- Cloud-native development is a specific type of cloud development that more references what, rather than how, software is developed. The term originates from the Kubernetes and Cloud Native Computing Foundation (CNCF) communities and defines cloud-native development as container-based, dynamically orchestrated software development that uses a microservices-based architecture. There is an argument that ‘container-native development’, is a better, less ambiguous term than cloud-native.



Cloud Development

Cloud Native

- Cloud-native application architecture, which is based on dynamically orchestrated containers, mirrors the qualities of elastic scalability and high availability of cloud infrastructures. Standard cloud development architectures don't use containers and rely on the cloud-infrastructure itself for scalability and high availability.



Cloud Native Development

Key Components in Cloud Native Development

- Containers
- Micro services
- Serverless
- CI/CD pipeline
- DevOps processes

REFER : <https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/definition>

Cloud Native Development

Benefits of Cloud Native Development

- Innovation
- Faster Releases
- Scalability and Elasticity
- Efficiency

Cloud Native Development

Challenges

- Culture Change
- Skills Gap
- Legacy Infrastructure and Applications
- Security



Summary

What have I learned?



Summary

What is
Cloud Computing?

SPOT TEST

Online Quiz

A photograph of a woman from behind, wearing a striped shirt and carrying a backpack, interacting with a digital kiosk mounted on a wall. The kiosk screen displays the question "How was your day?" with a five-star rating scale below it. A hand is pointing at the screen. The scene is set in a modern interior with a potted plant in the foreground.

Please Provide
Us Feedback?

Thank you!

#MOMENTOFSERVICE





© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.

Not Only SQL

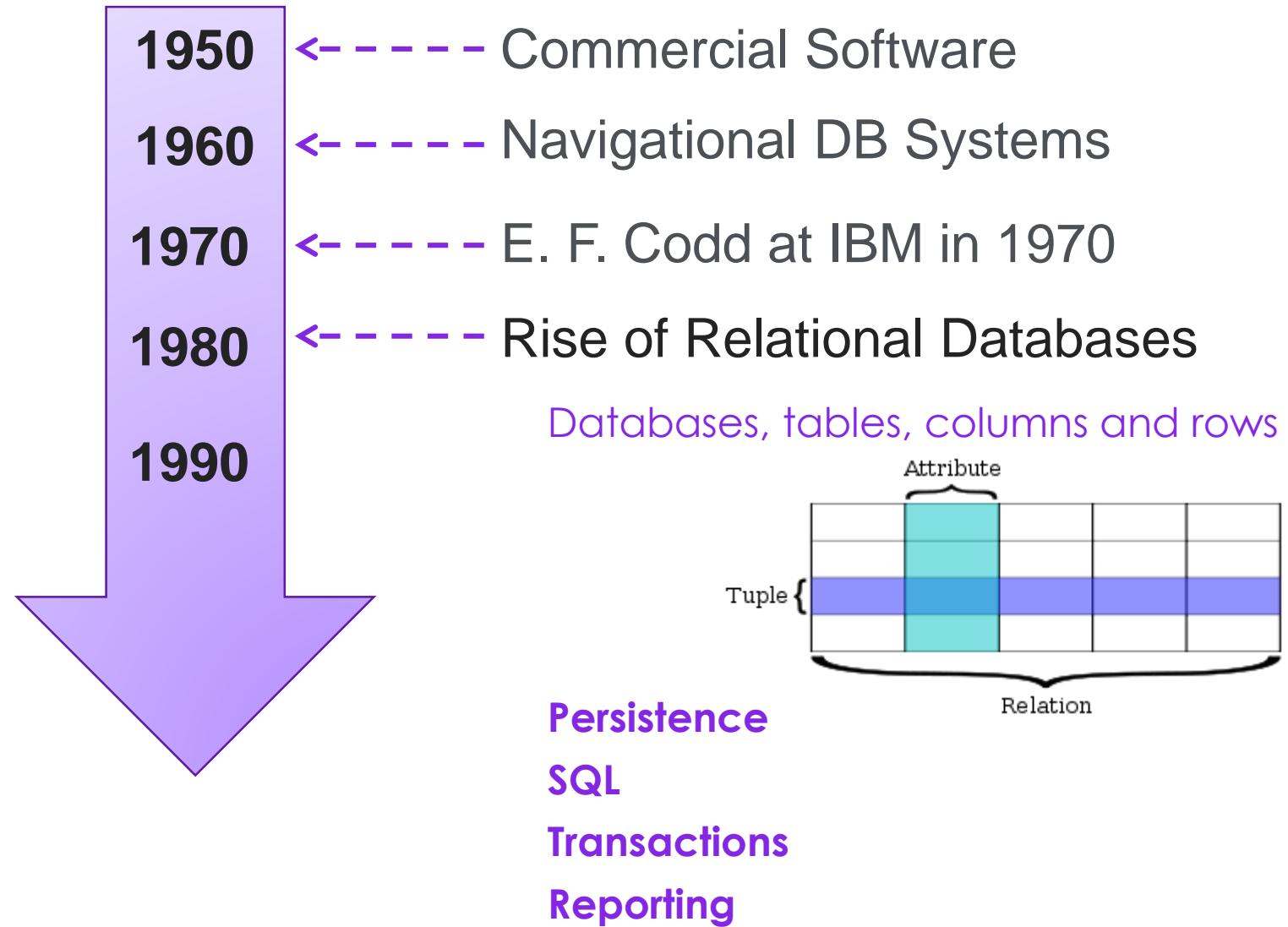
Enterprise Application Development

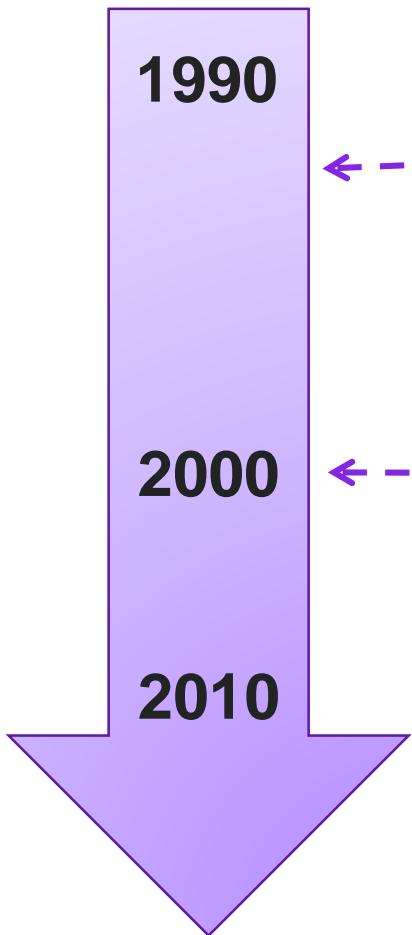
IFS Academy

Content

- * History of the database evaluation
 - * What is SQL
 - * What is NOSQL
 - * NoSQL vs SQL
 - * CAP & ACID
-
- * Types of NoSQL databases
 - * Advantage Disadvantage of NoSQL

History of the database evaluation

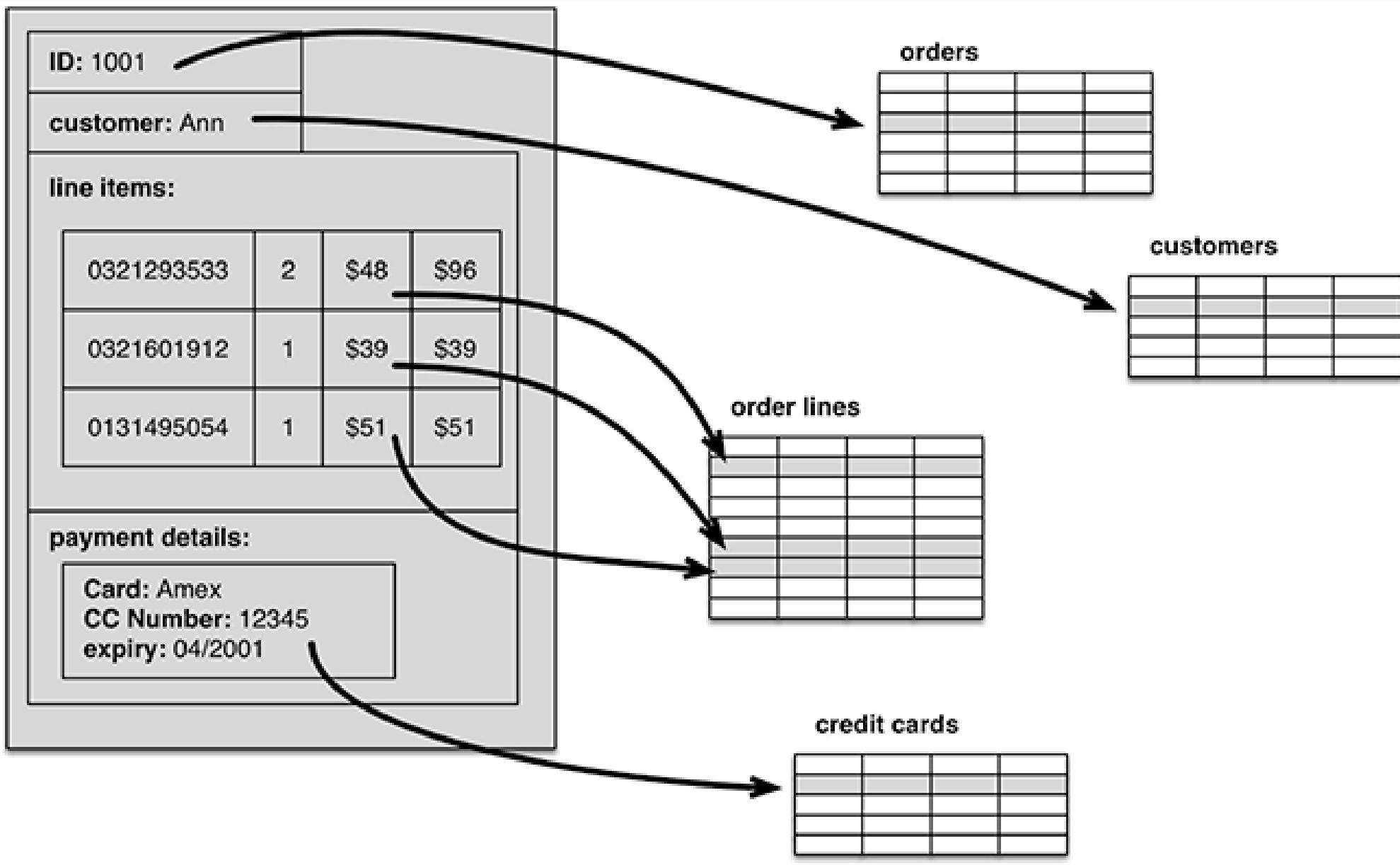




Relational Dominance

Use of relational databases became dominant and widespread.

NoSQL



Why we need NoSQL

What Changed this?



Enter NoSQL

RDBMS doesn't quite fit for **some** requirements

- Rigid Schema

- Not Scale friendly

Google and Amazon decided to make their own stuff (aka BigTable and S3 Storage) to meet their own unique needs

- Flexible Schema**

- Distributed**

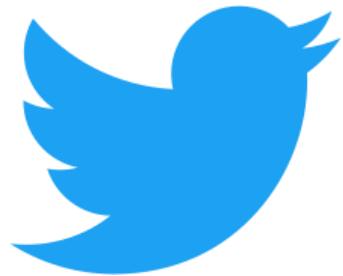
- Scalability**

- Control over performance characteristics**

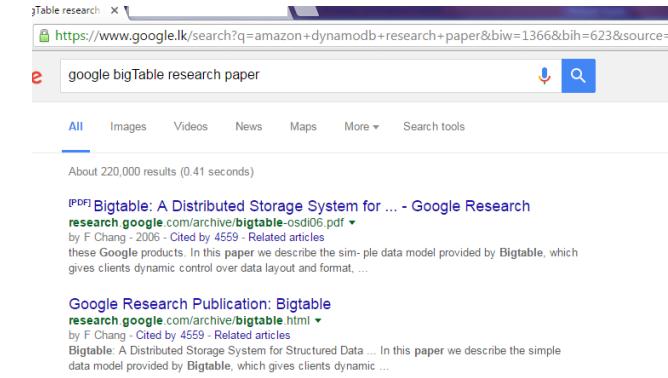
- High availability**

- Low Latency**

- Cheap**



A screenshot of a Twitter profile for the account @NoSQLDigest. The profile picture is a red and yellow parrot. The bio reads: "NoSQL Digest of tweets." The stats show 820K tweets, 14 following, and 14.9K followers. There is a "Follow" button. Below the profile, a tweet from Microsoft Azure (@Azure) is shown, liked 29 times. The tweet reads: ".@rimmanehme shares her thoughts on #Azure #DocumentDB & #NoSQL. Read on @Medium: msft.it/6013B4o81".



A screenshot of a Google search results page for the query "google bigTable research paper". The search bar shows the query. The results include a link to a PDF titled "Bigtable: A Distributed Storage System for ... - Google Research" and a link to a Google Research publication titled "Google Research Publication: Bigtable". Both results mention F Chang and a citation count of 4559.

NoSQL vs SQL

SQL

A common (for the most part) query language for RDBMS

Enables unrestrained queries against normalized data

Geared towards joins, filters, and aggregations

Relies heavily on the database server's query optimization

Sometimes requires the involvement of a DBA

What is NoSQL?

Stands for **Not Only SQL**

Class of non-relational data storage systems

Usually do not require a fixed table schema nor do they use the concept of joins

Natively cluster friendly(horizontal clustering)

All NoSQL offerings relax one or more of the ACID properties

SQL & Consistency

Relational Databases = **ACID** (**A**tomicity, **C**onsistency, **I**solation, **D**urability)

Atomicity – Either all the tasks within a transaction are performed or none of them are.

Consistency – Data should remain consistent at the beginning of a transaction as well as at the end of it.

Isolation – No transaction has access to any other transactions.

Durability – Once a transaction is complete, it will persist in the database and cannot be undone.

ACID is important but in specific cases

Banking, finance, safety systems, etc.

The kinds of systems that people were building with computers 30 years ago (and today)

CAP Theorem

ACID

A DBMS is expected to support “ACID transactions” processes that are:

Atomicity: either the whole process is done, or none is

Consistency: only valid data are written

Isolation: one operation at a time

Durability: once committed, it stays that way

CAP

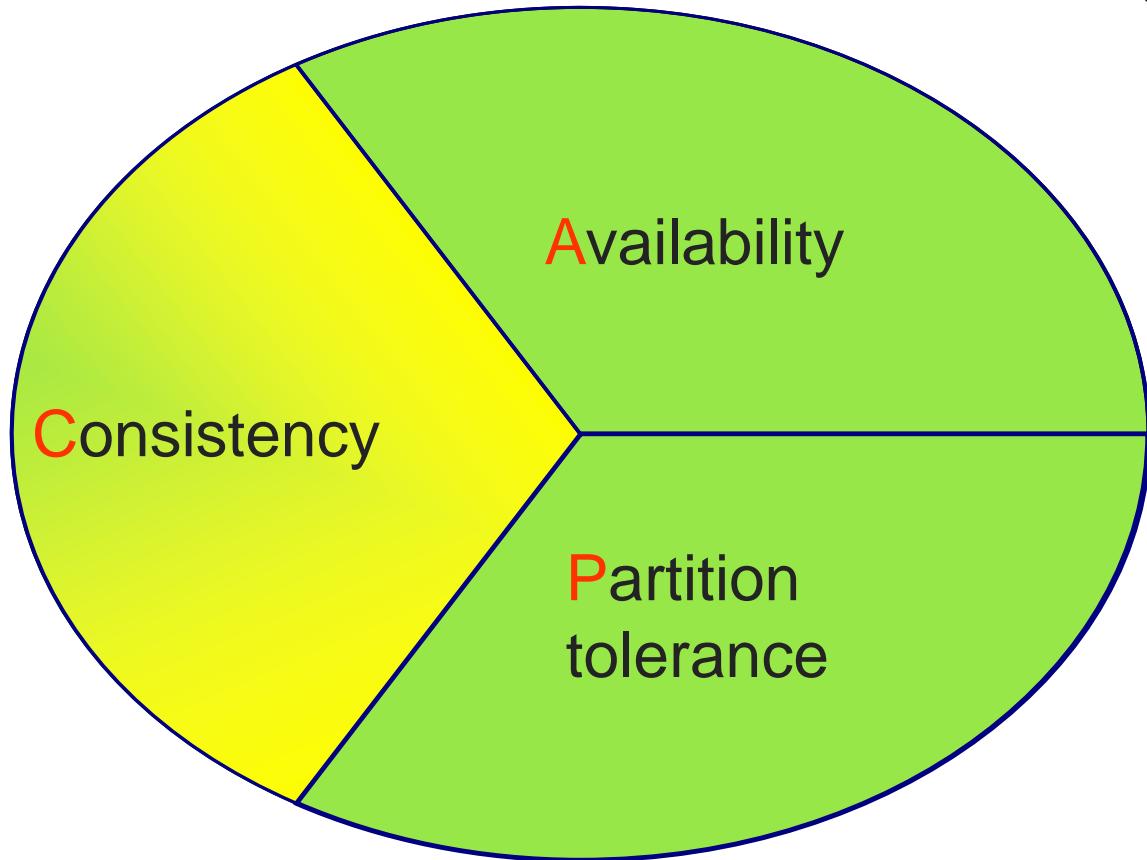
Consistency: all data on cluster has the same copies

Availability: cluster always accepts reads and writes

Partition tolerance: guaranteed properties are maintained even when network failures prevent some machines from communicating with others

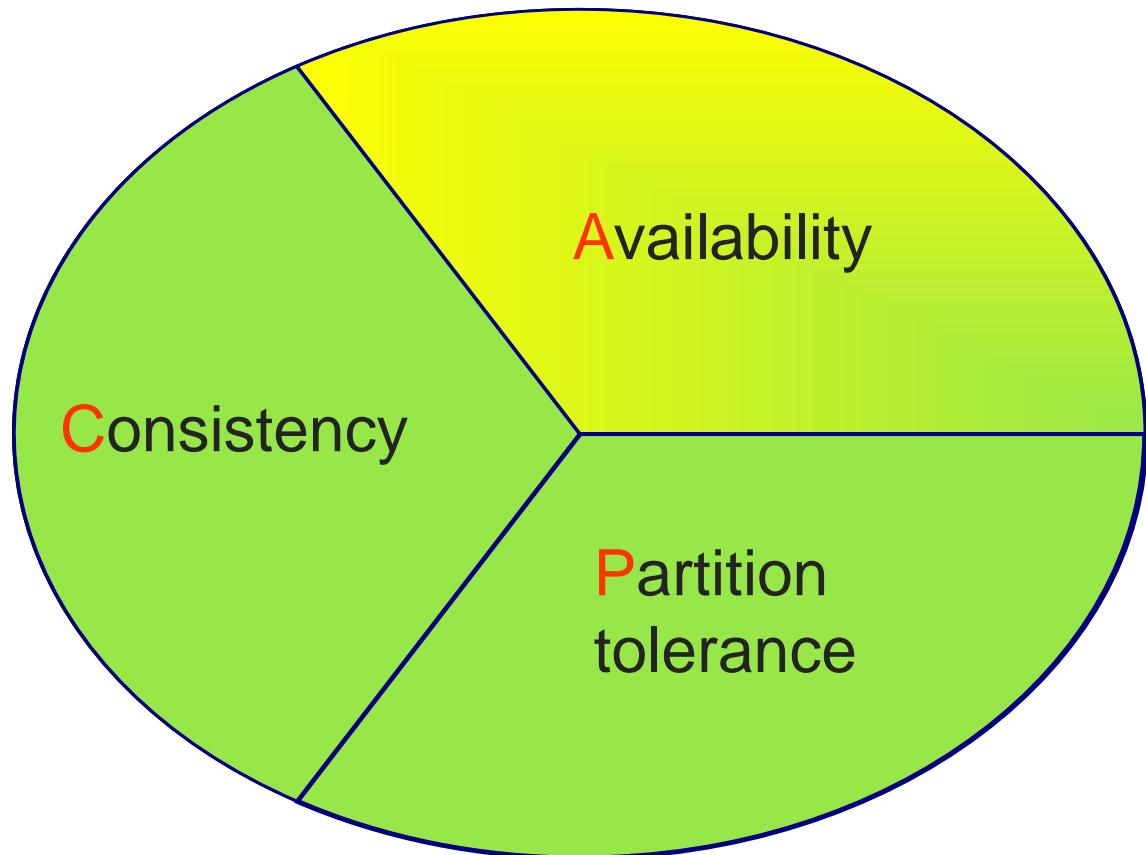
Can pick any of two

CAP Theorem



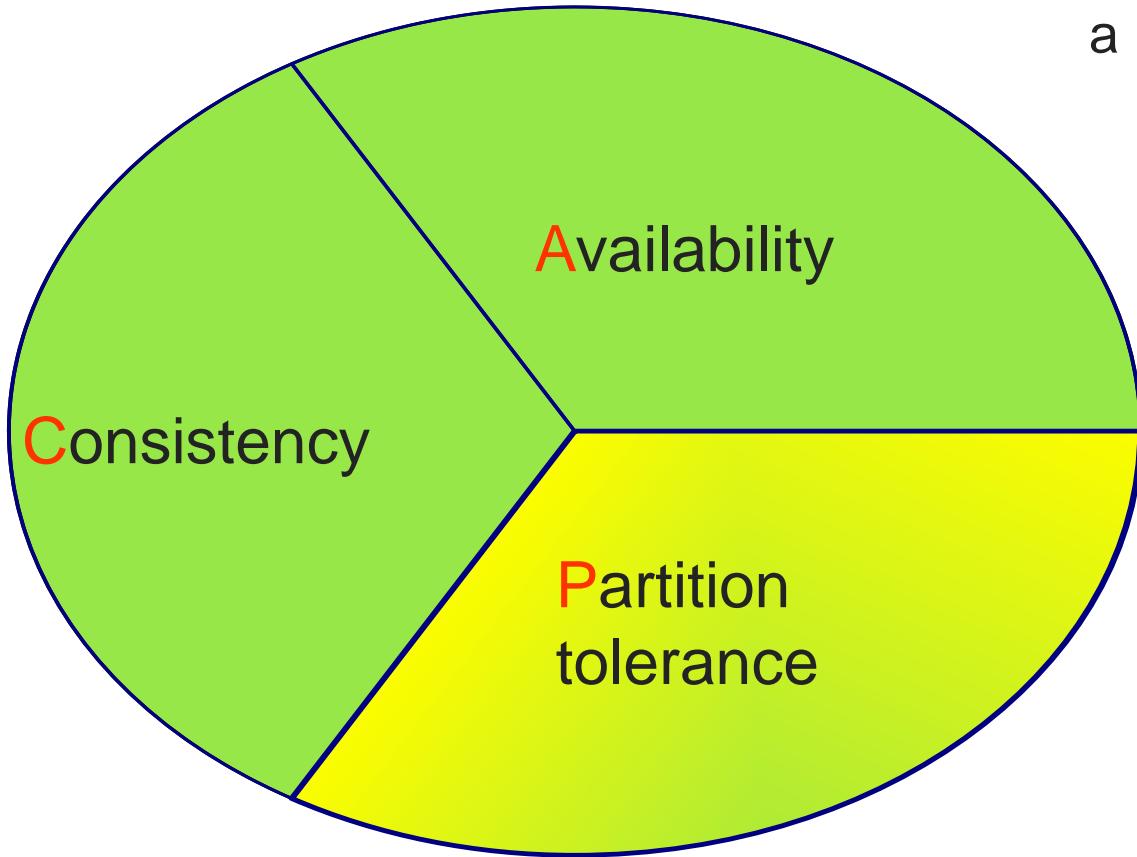
All client always have the same view
of the data

CAP Theorem

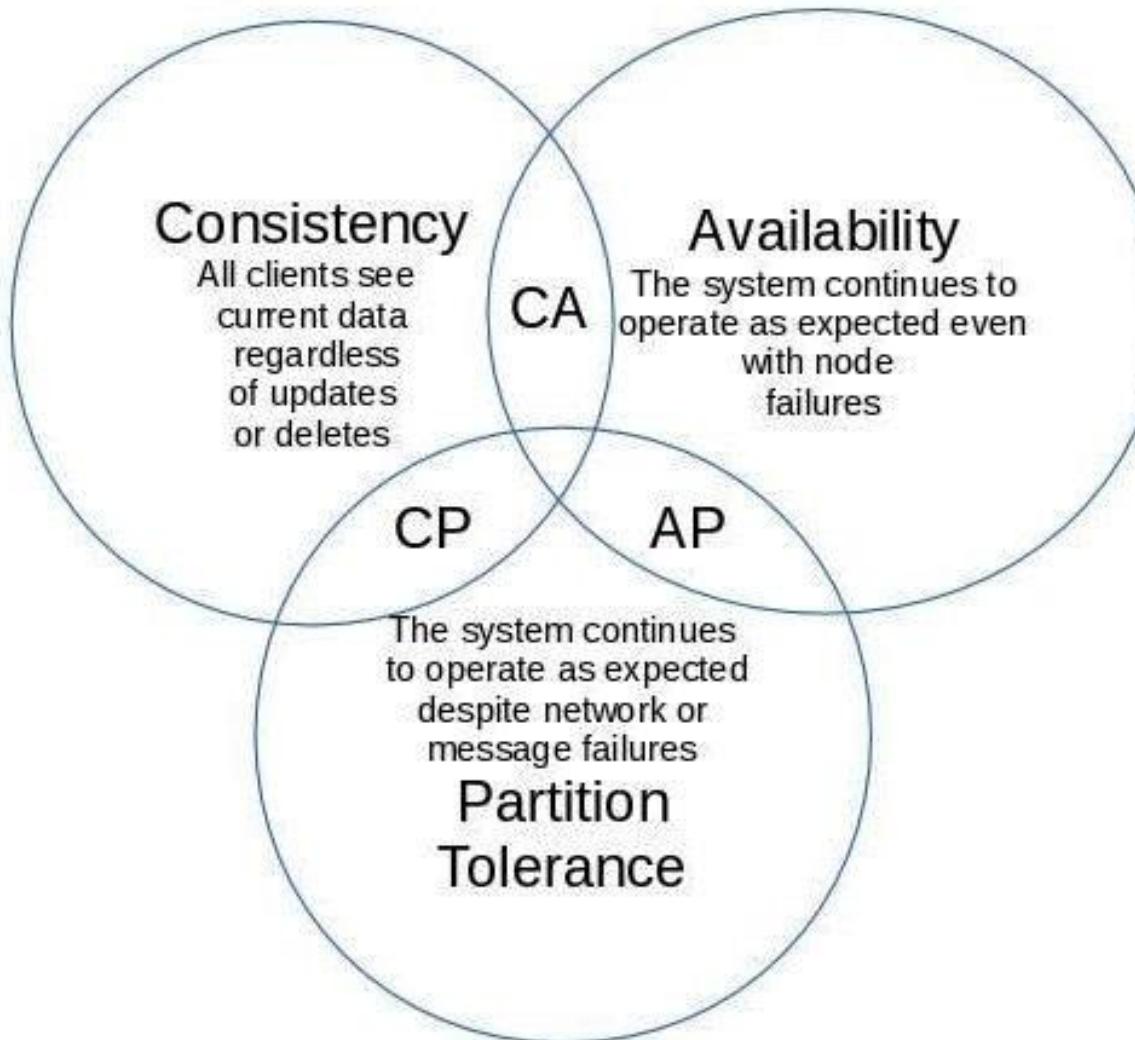


Each client always can read and write.

CAP Theorem



A system can continue to operate in the presence of a network partitions



RDBMS systems are designed for Consistency and Availability

NoSQL are more towards Availability and Partition Tolerance

What is NoSQL?

Basically a large serialized object store*

(mostly) retrieve objects by defined ID

In general, doesn't support complicated queries*

Doesn't have a structured (or any!) schema*

Designed to be distributed (cloud-scale) out of the box

Because of this, drops the ACID requirements

Any database can answer any query

Any write query can operate against any database and will “eventually” propagate to other distributed servers

* Dependent on vendor

NoSQL database types



Types of NoSQL databases

Type	Description	Examples
Key-value Stores	Simplest NoSQL databases. Every single item in the database is stored as an attribute name (or "key"), together with its value	Riak Voldemort Redis
Document Databases	Pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents	MongoDB ClusterPoint CouchDB MarkLogic
Graph Stores	Used to store information about networks, such as social connections	Neo4J Hyper, GraphDB OrientDB
Wide-column Stores	Optimized for queries over large datasets, and store columns of data together, instead of rows	Cassandra Hbase BigTable

Key-value stores

Simplest of NoSQL databases.

Stores data against a unique key.

Key is generally a string, and the value is stored as a blob.

Simple to build, easy to scale and usually have great performance.

Commonly used for managing user profiles, user sessions and shopping carts.

Most of the Key/Value stores expose some variation on the following API:

```
void Put(string key, byte[] data);  
byte[] Get(string key);  
void Remove(string key);
```

Key-value stores

Data can be stored in distributed nodes and queried using the key.

app_setting_width	450
user1923_color	Red
user1923_age	18
user3371_color	Blue
user4344_color	Brackish
user1923_height	6' 0"
user3371_age	34
error_msg_457	There is no file %1 here
error_message_1	There is no user with %1 name
1923_name	Jim
user1923_name	Jim Smith
user1923_lname	Smith
Application_Installed	true
log_errors	1
install_path	C:\Windows\System32\Restricted
ServerName	localhost
test	test
test1	test
test123	Brackish
devonly	
wonderwoman	
value	key

Document databases

Similar to key-value stores with one major difference.

Data is stored in an open format rather than a blob.

The format can be XML, JSON, Binary JSON (BSON), etc.

Allows server-side operations on data, and easy to create tools to manipulate data.

Document databases

terminology

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key <code>_id</code> provided by mongoDB itself)

A sample document

```
{  
    _id: ObjectId('7df78ad8902c'),  
    title: 'MongoDB Overview',  
    description: 'MongoDB is no sql database',  
    by: 'tutorials point',  
    url: 'http://www.tutorialspoint.com',  
    tags: ['mongodb', 'database', 'NoSQL'],  
    likes: 100,  
    comments: [  
        {  
            user: 'user1',  
            message: 'My first comment',  
            dateCreated: new Date(2011, 1, 20, 2, 15),  
            like: 0  
        },  
        {  
            user: 'user2',  
            message: 'My second comments',  
            dateCreated: new Date(2011, 1, 25, 7, 45),  
            like: 5  
        }  
    ]  
}
```

NoSQL & Consistency

- NoSQL Databases = **BASE** (Basically Available, Soft State, Eventual Consistency)

Basically Available – There will be a response to any request, but it could be a failure.

Soft State – The state of the system could change over time.

Eventual Consistency – The system will eventually become consistent when it stops receiving input.

Advantages of NoSQL

- Flexible data model
- Support for semi-structured and unstructured data
- Scalability
- Replication and high availability
- Low cost

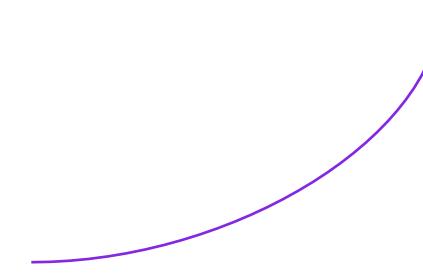
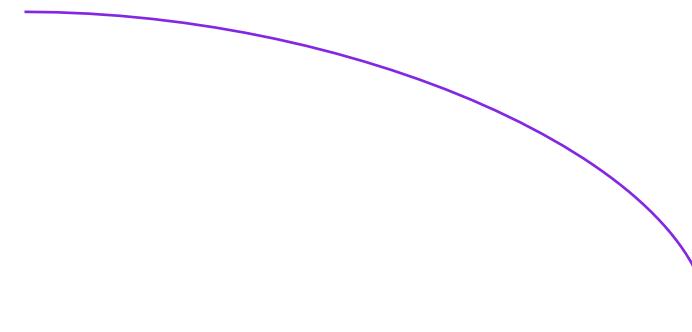
Disadvantage of NoSQL

- Decisions - What is the appropriate NoSQL Database
- Organizational Changes - Relational DBAs will not like it
- Immature - Knowledge, Experience, Tools...
- Eventual consistency

Easy to develop

NoSQL

Large amount of data



Thank you!

#MOMENTOFSERVICE





© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.

MongoDB

Enterprise Application Development

IFS Academy

MongoDB

How to install and use MongoDB

Standalone Installation

Managed software as a service - MongoDB Atlas

MongoDB Community

MongoDB Enterprise



mongoDB®

MongoDB

MongoDB commonly used developer tool set

- MongoDB Community Edition

- MongoDB Shell – Connect and work with MongoDB through a command line interface

- MongoDB Compass – GUI tool for querying, optimizing, and analyzing your MongoDB data



mongoDB®

MongoDB



```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
-----
test> show dbs
Actors      72.00 KiB
admin       40.00 KiB
config      72.00 KiB
local       72.00 KiB
simple_db   60.00 KiB
test> use Actors
switched to db Actors
Actors> show collections
fighters
Actors> db.fighters.insertOne( { firstname: "John" , lastname: "Connor" } )
{
  acknowledged: true,
  insertedId: ObjectId("62f4e3a745c3fb4cc3cfb685")
}
Actors> -
```

MongoDB

The screenshot shows the MongoDB Compass interface connected to localhost:27017. The left sidebar displays the database structure with 5 DBs and 5 Collections. The 'fightes' collection under the 'Actors' database is selected. The main panel shows the 'Documents' tab for the 'Actors.fightes' collection, which contains two documents. The first document has the following fields:

```
_id: ObjectId('62f22363288a5049420e4ab4')
firstname: "John"
lastname: "Connor"
Gender: "male"
```

The second document has the following fields:

```
_id: ObjectId('62f2281b288a5049420e4ab5')
firstname: "Sarah"
lastname: "Connor"
Gender: "female"
```

At the bottom, there is a MONGOSH prompt: >_MONGOSH

MongoDB

MongoDB Driver – A driver is needed to connect a MongoDB database with a programming language (JAVA C# etc.)

The screenshot shows the NuGet package manager interface. The top navigation bar has tabs: 'Browse' (underlined), 'Installed', and 'Updates 8'. Below the tabs is a search bar containing 'Mongodb.driver' with a clear button 'x', a refresh icon, and a checkbox 'Include prerelease'. The main area displays three packages:

Package	Downloads	Version
MongoDB.Driver by MongoDB Inc.	93.2M	2.17.1
MongoDB.Driver.Core by MongoDB Inc.	96.9M	2.17.1
MongoDB.Driver.GridFS by MongoDB Inc.	5.57M	2.17.1

Each package entry includes a green checkmark icon and a brief description: 'Official .NET driver for MongoDB.' for the first, 'Core Component of the Official MongoDB .NET Driver.' for the second, and 'GridFS Component of the Official MongoDB .NET Driver.' for the third. The first package is highlighted with a blue border.



© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.

BIG DATA



SUPUL JAYAWARDANE

WHAT IS BIG DATA

Collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications.

Extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behaviour and interactions.

MAGNITUDE OF DATA

By 2020, about 1.7 megabyte of new information will be created every second for every human being on the planet.

By 2020, accumulated digital universe of data will grow from 4.4 zettabytes in 2015 to around 44 zettabytes, or 44 trillion gigabytes.

By 2015, we performed 40,000 Google queries every second which makes it 3.5 searches per day and 1.2 trillion searches per year.

1.5 billion smartphone were shipped in 2016 - all packed with sensors capable of collecting all kinds of data, not to mention the data the users create themselves.

Between 2015-2020, it is expected that 50 billion smart connected devices will come online, all developed to collect, analyze and share data.

By 2015, less than 0.5% of all data is ever analysed and used, just imagine the potential !!

MORE FACTS

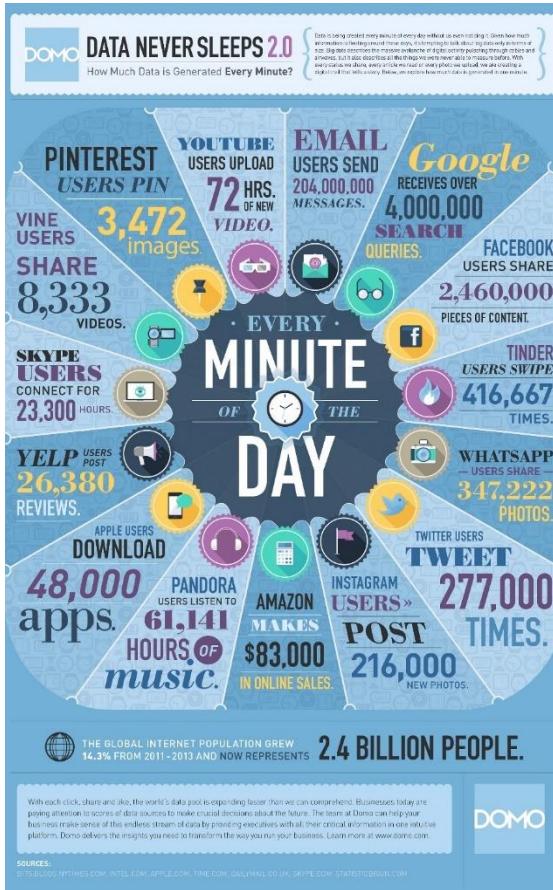
**Google processes 24 PB a day
(2009)**

**Wayback Machine has 3 PB +
100 TB/month (2009)**

**Facebook has 2.5 PB of user
data + 15 TB/day (2009)**

**eBay has 6.5 PB of user data
+ 50 TB/day (2009)**

**eBay has 6.5 PB of user data
+ 50 TB/day (2009)**



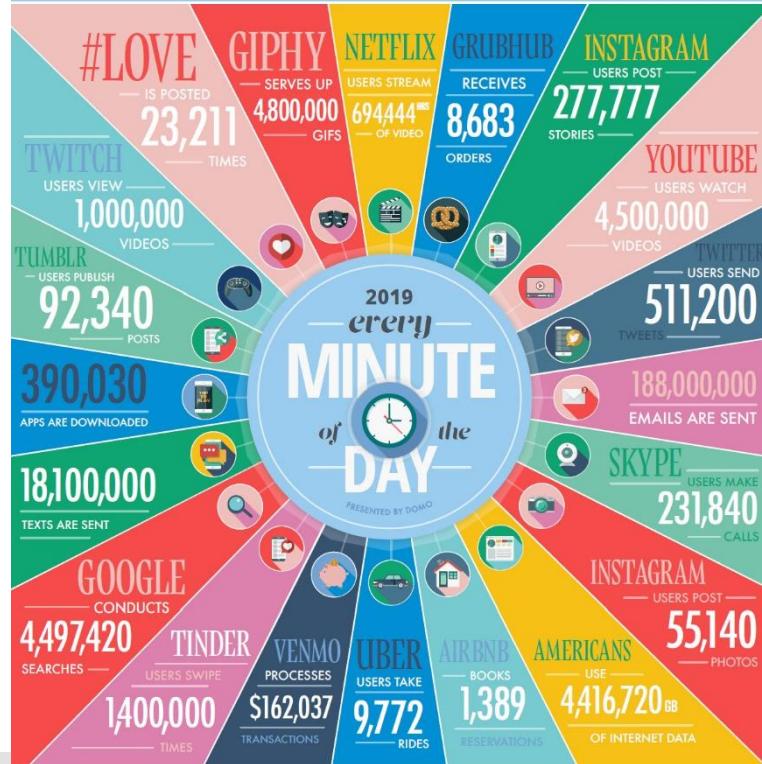
Source: <https://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/>

DOMO

DATA NEVER SLEEPS 7.0

How much data is generated *every minute*?

There's no way around it: big data just keeps getting bigger. The numbers are staggering, and they're not slowing down. By 2020, there will be 40x more bytes of data than there are stars in the observable universe. In our 7th edition of Data Never Sleeps, we bring you the latest stats on how much data is being created in every digital minute.



SOURCES: STATISTA, INTERNET LIVE STATS, EXPANDED RAMBLINGS, NATIONAL ASSOCIATION OF CITY TRANSPORTATION OFFICIALS, WIRED



DATA GENERATION

SOCIAL MEDIA

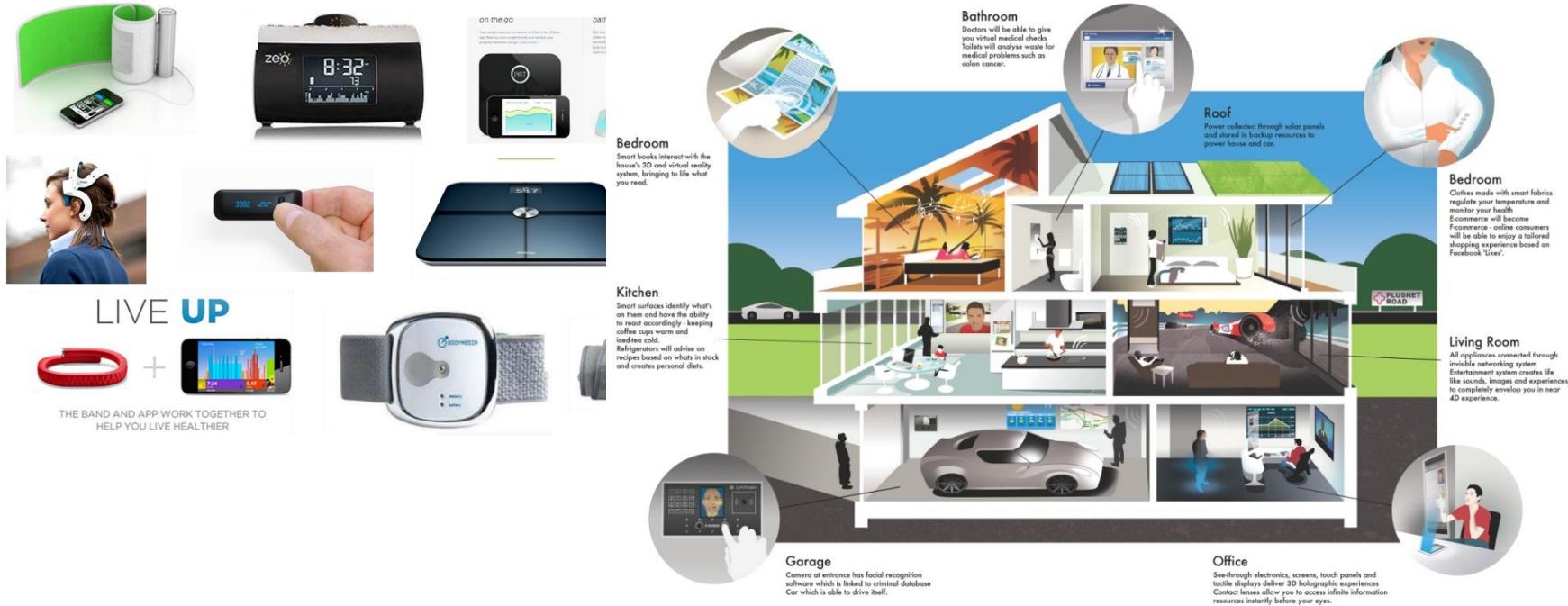


MOBILE DEVICES



SENSOR TECHNOLOGY AND NETWORKS

PROCESSING POWER AND SENSORS HAVE BECOME AFFORDABLE



NATURE OF DATA

DATA MODEL

THEN

Few companies were generating data and all others were consuming

Traditional media – News papers, TV, Radio channels

Government

NOW

All of us are generating data and all of us are consuming

- Social media – FB, Instagram
- Crowd sourcing – Google maps

DATA TYPES

● Structured Data

- Data is organized according to a specific structure or schema.
- Eg: Data stored in a relational database.

● Unstructured Data

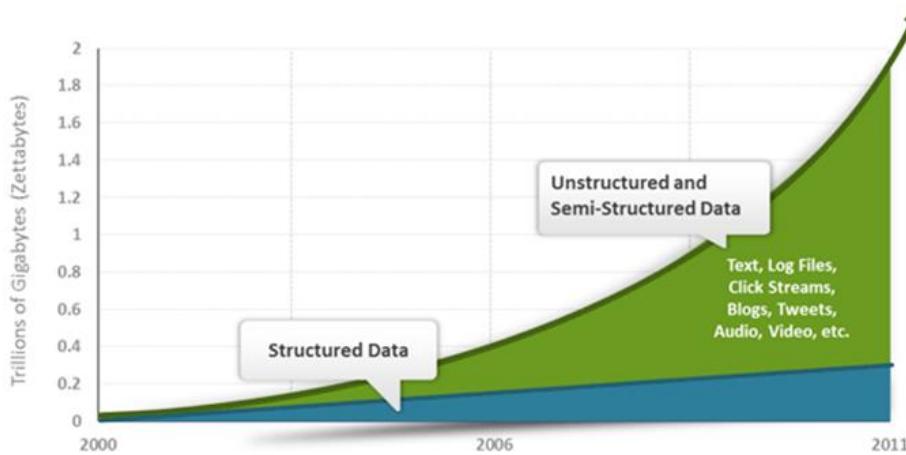
- Data that does not have any consistent organization, associated metadata or schema.
- Eg: Certain documents, images, audio and video files

● Semi-Structured Data

- Data that has some form of organization.
- Eg: Server logs, XML / email documents, sensor data.

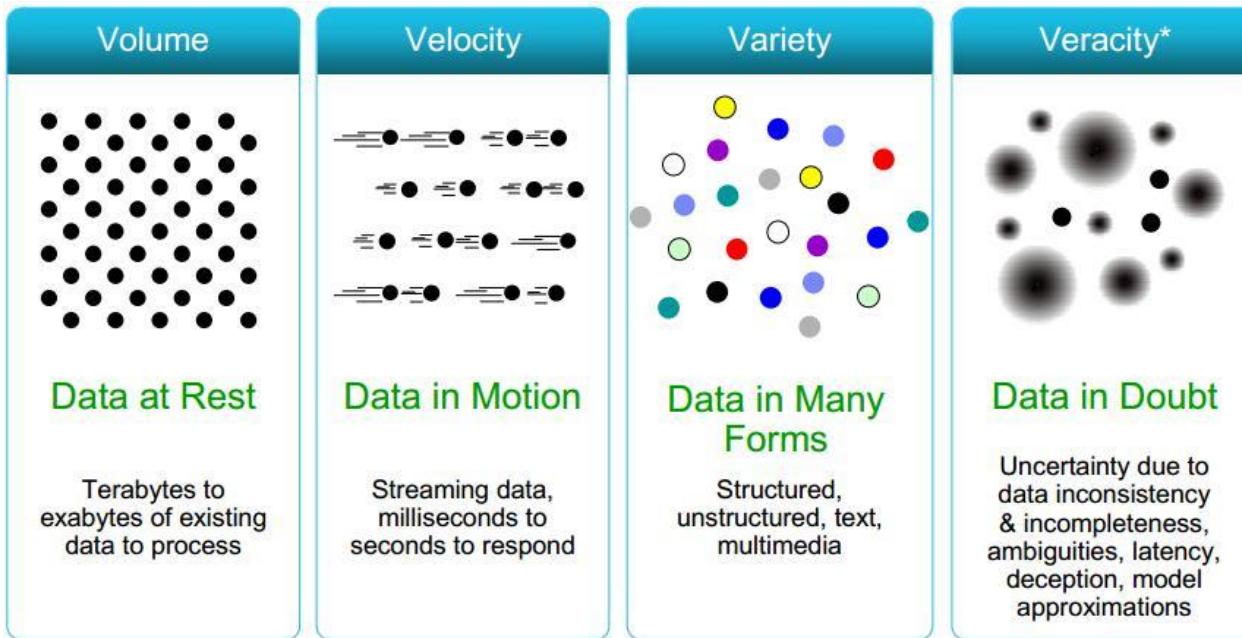
BIG DATA?

- Data volume has grown exponentially over the past decade.



80% of the world's data is **unstructured**

CHARACTERISTICS



40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by
2020, an increase of 300
times from 2005



Volume SCALE OF DATA

It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day



The FOUR V's of Big Data

From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?

As a leader in the sector, IBM data scientists break big data into four dimensions: **Volume**, **Velocity**, **Variety** and **Veracity**.

Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States.



The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session



Velocity ANALYSIS OF STREAMING DATA

By 2016, it is projected
there will be
**18.9 BILLION
NETWORK CONNECTIONS**
— almost 2.5 connections
per person on earth



As of 2011, the global size of
data in healthcare was
estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]



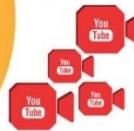
Variety DIFFERENT FORMS OF DATA

**30 BILLION
PIECES OF CONTENT**
are shared on Facebook
every month



By 2014, it's anticipated
there will be
**420 MILLION
WEARABLE, WIRELESS
HEALTH MONITORS**

4 BILLION+
HOURS OF VIDEO
are watched on
YouTube each month



400 MILLION TWEETS
are sent per day by about 200
million monthly active users



**1 IN 3 BUSINESS
LEADERS**
don't trust the information
they use to make decisions



Veracity UNCERTAINTY OF DATA

**27% OF
RESPONDENTS**
in one survey were unsure of
how much of their data was
inaccurate

BIG DATA ANALYTICS

Big data analytics is the process of examining **large datasets** to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information.

Source: <http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics>

ML/AI AND BIG DATA

- Machine Learning and AI is becoming increasingly popular when generating output from Big Data.
- AI/ML technologies are used to find patterns more effectively and ML is used for predictive analytics using big data

BIG DATA ANALYTICAL PLATFORMS



BIG DATA



• SUMMARY

ENTERPRISE APPLICATION SOFTWARE (EAS)



SUPUL JAYAWARDANE

supul.Jayawardane@ifs.com

WHAT IS ENTERPRISE APPLICATION SOFTWARE (EAS)?

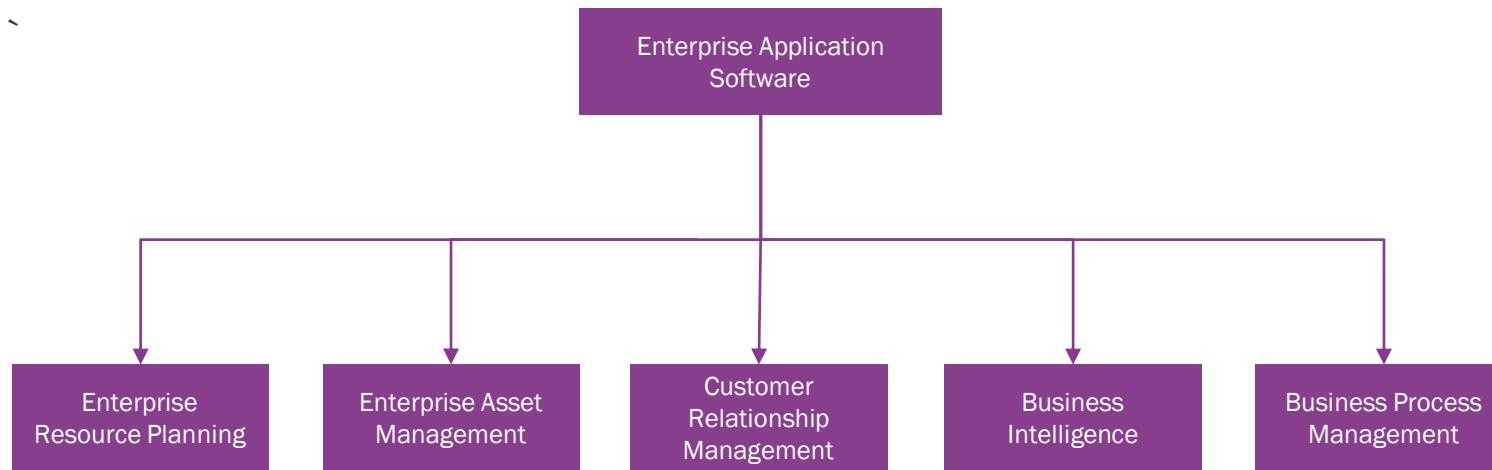
DEFINITION

Enterprise Application Software is a computer software used to satisfy the needs of an organization rather than individual users. Such organizations would include businesses, schools, interest-based user groups, clubs, charities, or governments. Enterprise software is an integral part of a (computer-based) information system.

WHAT EAS DOES

- EAS attempts to **integrate** all departments and functions across a company onto a single computer system that can serve all those different departments' particular needs.
- Automates the tasks involved in performing a business process

ENTERPRISE APPLICATION SOFTWARE TYPES



EAS IN BUSINESS



EAS IS BEING USED TO BUILD THE ROYAL NAVY'S TWO LARGEST EVER AIRCRAFT CARRIERS, THE HMS QUEEN ELIZABETH AND HMS PRINCE OF WALES: **PROJECT WORTH £6.2BLN**

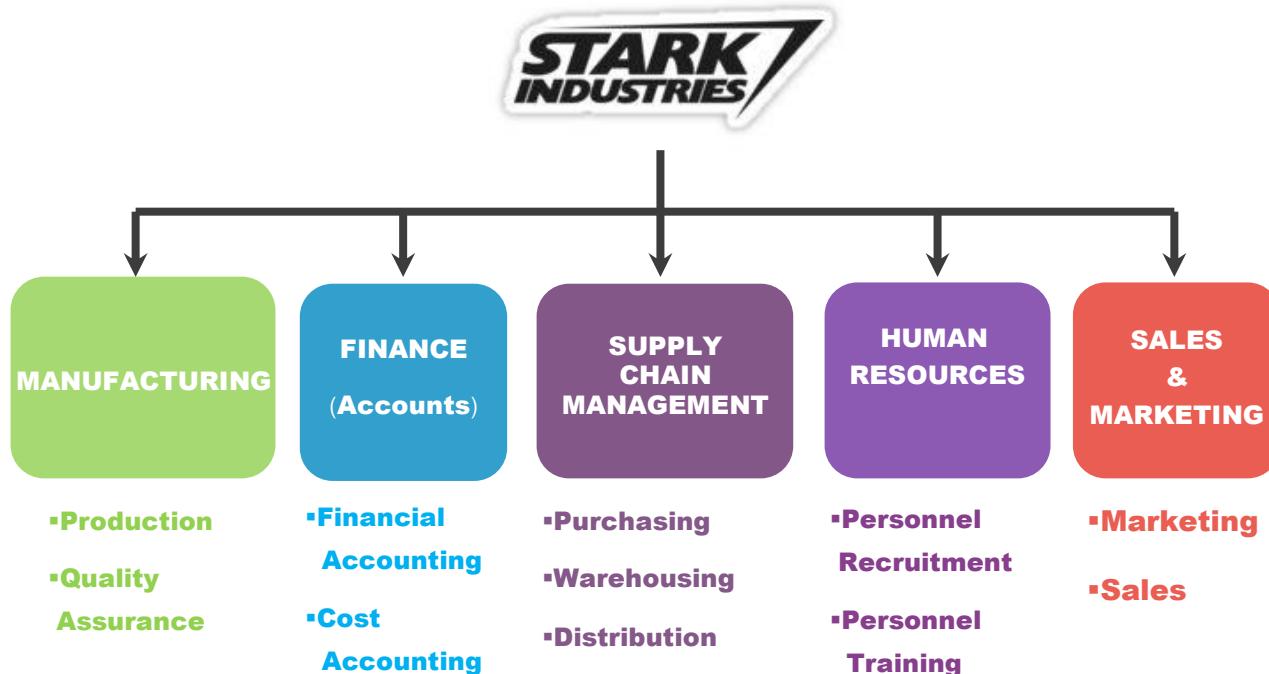
Image property of Navantia



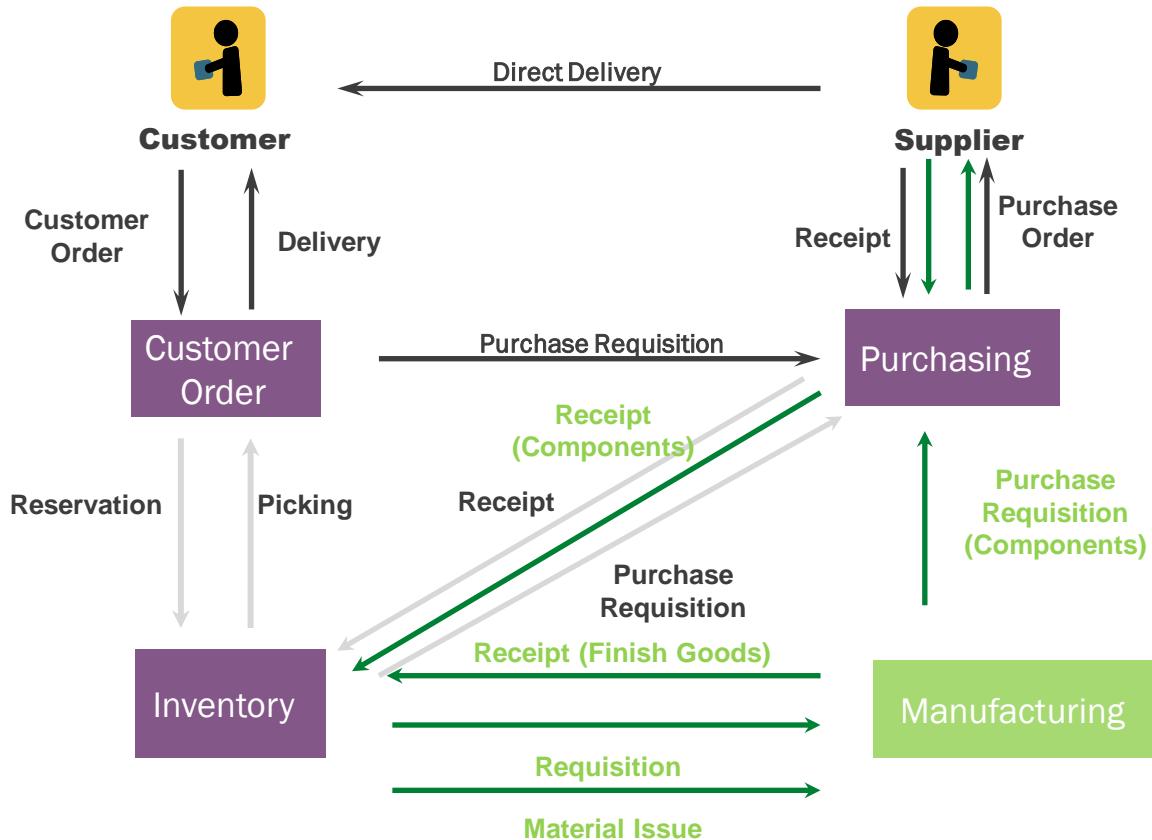
IFS IS BEING USED BY THE NORWEGIAN NAVY TO OPERATE VESSELS FOR **6 MONTHS IN FULLY AUTONOMOUS MODE** DURING NATO PEACEKEEPING MISSIONS

ERP SCENARIO

BUSINESS OVERVIEW



BUSINESS PROCESS (CLOSER TO REALITY)



EAM SCENARIO (NORWEGIAN ARMED FORCES)



Airforce and Navy

- F-16 Fighters (57)
- C-130J Transport (6)
- P3C/N Orion Surveillance (6)
- DA-20 Jet Falcon Electronic Warfare (3)
- Bell 412 Helicopters Tactical (18)
- Sea King Helicopters SAR (12)
- Lynx Helicopters Coastguard (6)
- SAAB Safari Trainer (15)

- Frigates (5)
- Corvettes (6)
- Submarines (6)



ENTERPRISE ASSET MANAGEMENT SCENARIO



NORWEGIAN ARMED FORCES

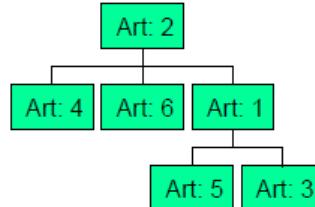
Norwegian Defence Logistics Organisation

DNF Facts

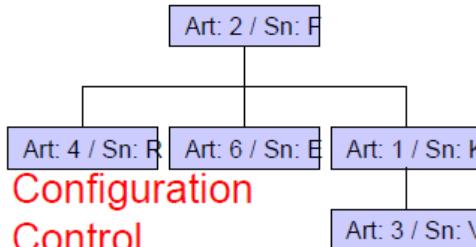
- Parts total: 50.000
- Documents: 300.000/+50.000
- Approval routing steps/ Work-flow: 1.250.000
- Allowed conf fregatt: 17.000
- Actual conf fregates: 10.500
- Taks in MP Fregates: 3.500
- # spareparts in warehouse land: 12.500
- # spareparts in ships store Fregates: 7.000
- Inventory transactions total: 280.000

Configuration Management

- Technical structure:
- Technical Work breakdown Structure (Configuration Item (CI))
 - Spare parts
 - Technical documentation



- Serial number structure:
- Serial numbers
 - Based on technical structure



Configuration Control

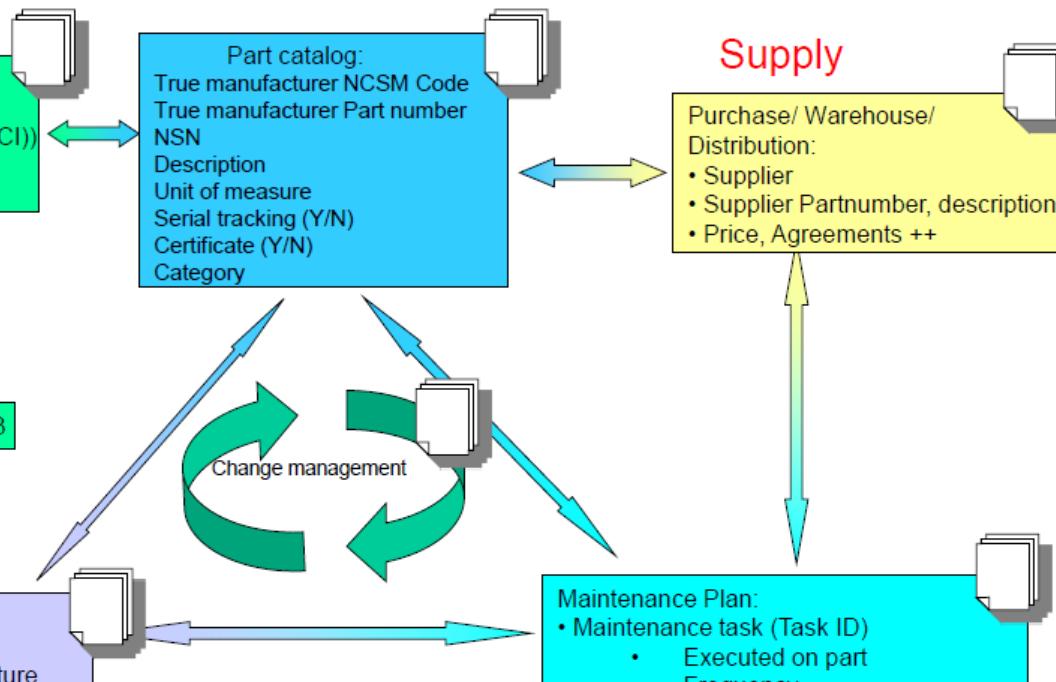
Functional relations – IFS Appl 7.3

Supply

- Purchase/ Warehouse/ Distribution:
- Supplier
 - Supplier Partnumber, description
 - Price, Agreements ++

- Maintenance Plan:
- Maintenance task (Task ID)
 - Executed on part
 - Frequency
 - Spare parts
 - Documentation
 - Services
 - Maintenance type
 - Level of maintenance (level 1-3)
 - Department (Platform/Weapons)
 - Man hours

Maintenance Management



TYPICAL USERS OF EAS SYSTEMS



DEVELOPING & BUYING EAS

DEVELOPING EAS

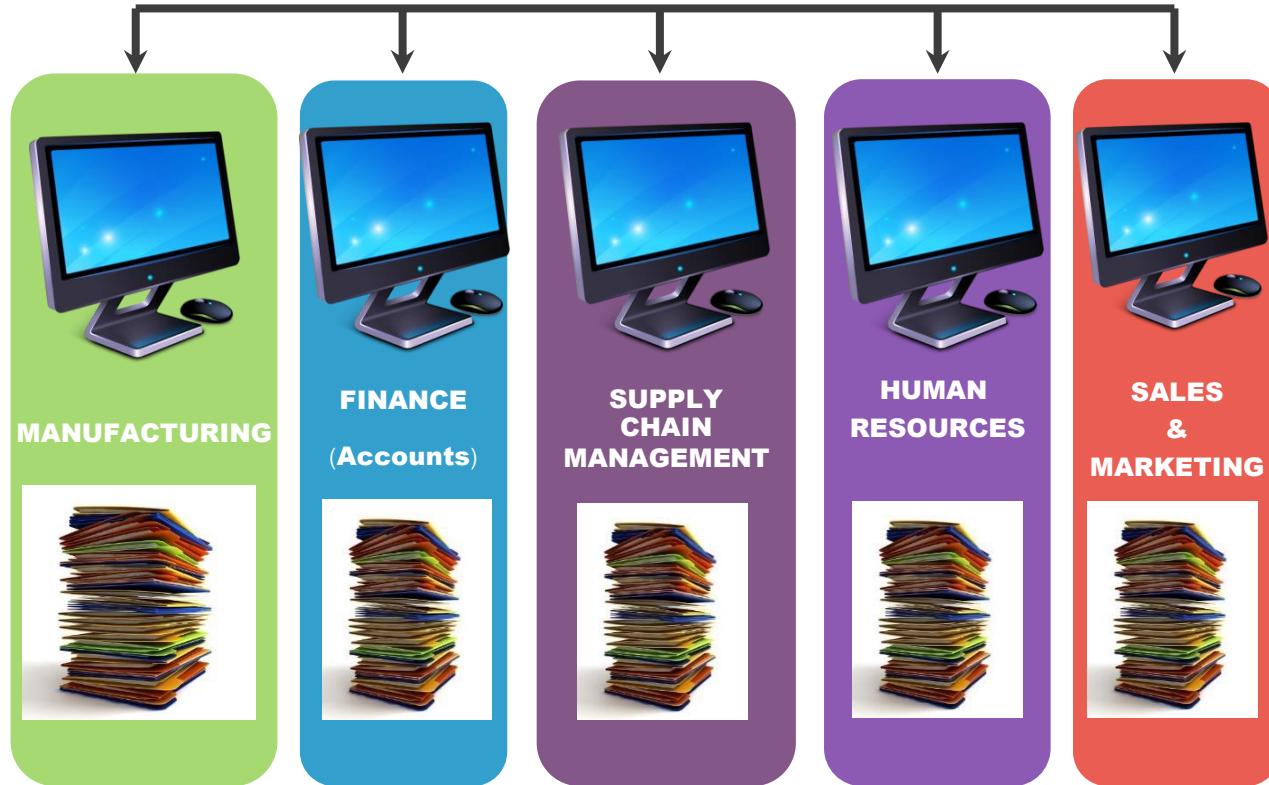
- Although EAS development is mostly similar to other software development, there are several factors that should be specially considered
 - Nature of the users (IT Literacy)
 - Software usage environment (office space, factory floor, aircraft hangar)
 - Regulatory requirements
 - Scalability
 - *Ease of maintenance*
 - *Reliability and performance*

BUYING EAS

- As a user, when buying an enterprise software, you should consider some of the following
 - Requirement fit
 - Licensing cost vs value
 - Performance
 - Ease of maintenance
 - Extensibility
 - Vendor reputation

HISTORY OF EAS

LIFE BEFORE EAS/ERP



LIFE BEFORE EAS/ERP

- Too many paperwork
- Information related to different business processes are isolated
- Lots of off the record interaction between employees leading to confusion
- Lots of inefficiencies and delays
- Computer software usage is limited to isolated departments and required redundant data entry

MODERN EAS TRENDS

CLOUD COMPUTING



Scalability

IOT

The **INTERNET** of **THINGS**

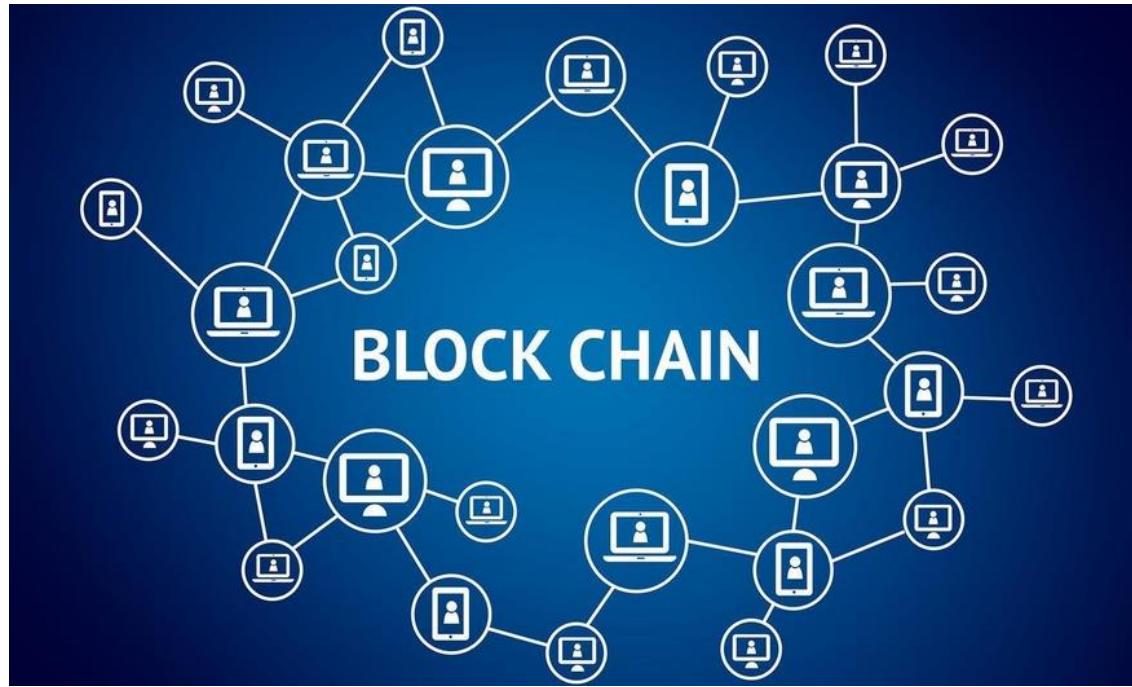


- Automated data capture
 - More data inputs
 - Automated trigger points
 - Enhanced integration
 - Minimal human interaction

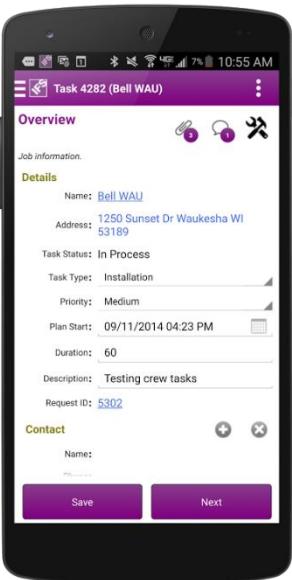
AI/MACHINE LEARNING



BLOCKCHAIN



MOBILE ACCESS



The tablet screen shows a 'Customer Order - D550341' page. The main area displays order details: Order No: D550341, Customer: HAKEXT02, Site: DEMO4, Coordinator: RES, Order Type: Sales, Currency: GBP, Status: Planned, Priority: 1, Reference: 01, Delivery Address: External customer 02, Document Address: 01, Wanted Delivery Date/Time: 2015-02-02 00:00:00. The 'Actions' overlay window is open, containing various buttons for order management: Release, Print Order Confirmation, Print Pro Forma Invoice, Cancel Order, Calculate Group Discount, Sales Promotions, Recalculate Commission, Calculate/Consolidate Freight, Calculate Earliest Delivery Date, Manual Credit Check, Create Advance Invoice, Connected Work Orders, Service Request, and Create Project. At the bottom of the tablet screen, there are tabs for Tasks, Attachments, Output, and Help.

EAS INDUSTRY FOCUS

- Automotive
- Chemical
- Pharmaceutical
- Consumer Goods
- Oil and Gas
- Aerospace & Defense

PROS & CONS

PROS

- A Totally integrated system
- The ability to streamline different processes and workflows
- The ability to easily share data across various departments in an organization
- Improved efficiency and productivity levels
- Better tracking and forecasting
- Lower costs
- Improved customer service
- Improved data analysis
- Helps in strategic decision making

CONS

- Customization in many situations is limited
- The need to re-engineer business processes
- EAS systems can be cost prohibitive to install and run
 - Training
 - Integration
 - Data Analysis and Conversion
- Technical support can be sloppy
- Security Issues
- ERP's may be too rigid for specific organizations that are either new or want to move in a new direction in the near future.

QUESTIONS?

Enterprise Mobile Applications

Extending Enterprise Applications to Mobile Devices

Anjula Priyanath
Senior Software Engineer



Contents

- 01 Mobile Applications
- 02 Introduction to Enterprise Applications
- 03 Extending Enterprise Applications to Mobile Devices
- 04 System Architecture

- 05 Application Architecture & Design
- 06 Application Model
- 07 Application Framework Components (A Proof of Concept)
- 08 Technology Stacks Available

Mobile Applications

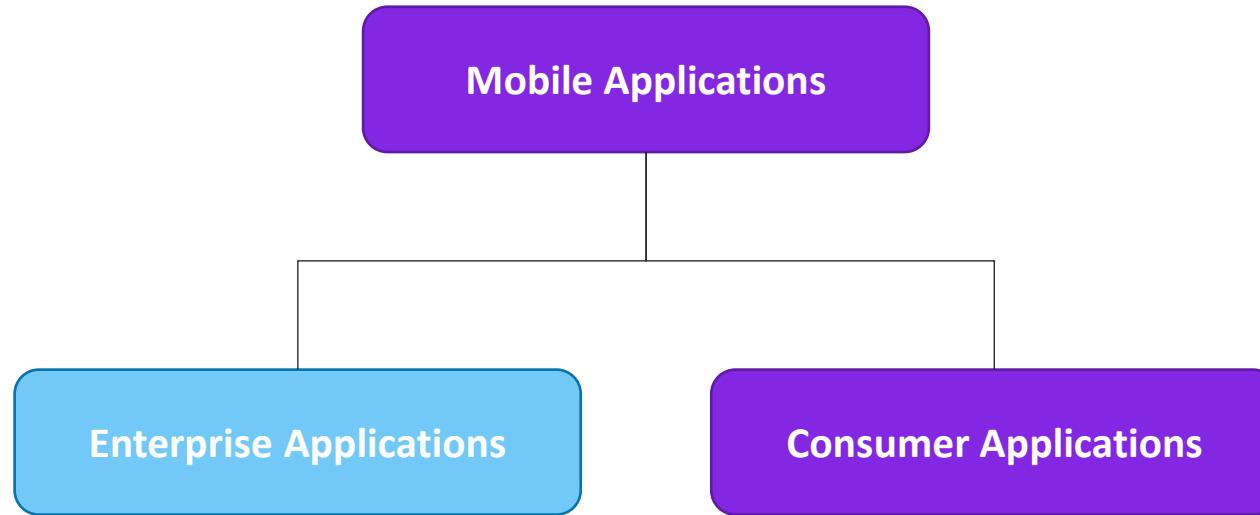
Mobile Applications

There are Many Types of Mobile Applications



Two Main Types of Mobile Applications

Based on Their Usage



Mobile applications integrated with back-end enterprise applications.

Stand-alone applications developed specifically for consumers.

Introduction to Enterprise Applications

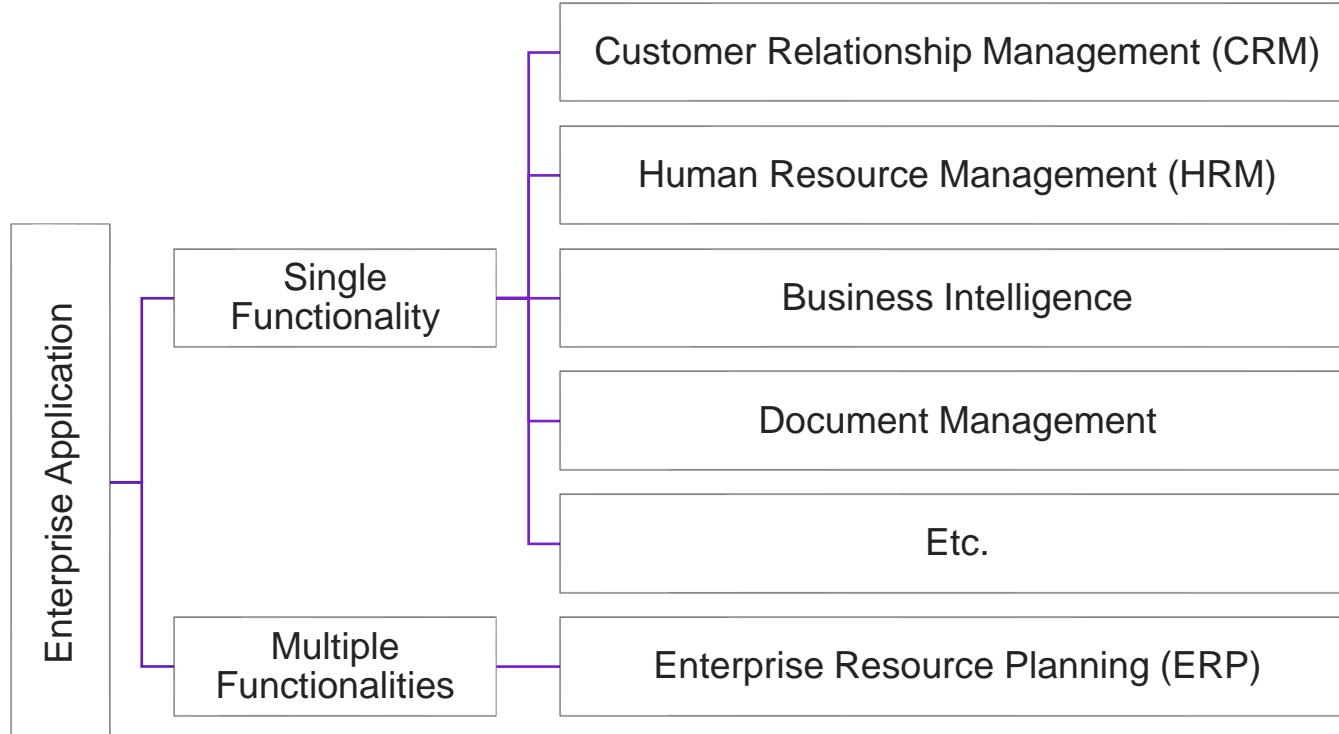
What is an Enterprise?

- An Enterprise can be considered as any Business Organization or Government Organization who manages many business operations.
- Few examples:
 - Manufacturing Companies (Vehicles, Furniture, Shoes etc.)
 - Rail & Transit Companies (Trains, Buses and Airlines)
 - Electricity Power Plant Operators
 - Military Organizations (Air Force, Army and Navy)
 - Maintenance Organizations (Maintaining Railways, Aircrafts etc)
 - Distributors
 - Satellite Television Broadcasters
 - And many others...



What is an Enterprise Application?

- A Software System that can be used to manage Business Operations of an Enterprise.
- There are mainly two types of Enterprise Applications:



Enterprise Application Segments

According to Functionality



ENGINEERING



MANUFACTURING



MAINTENANCE



DISTRIBUTION



CUSTOMER RELATIONSHIP
MGMT (CRM)

PLB	4.30	-0.31	-0.70	0.01
MAL	2.42	-0.01	-0.20	0.01
NE	5.62	-0.07	-0.20	0.01
MV	2.49	-0.02	-0.20	0.01
QTVV	1.12	-0.04	-0.20	0.01
HYOS	21.12	-0.11	-0.34	0.11
PLUG	26.37	-0.04	-0.34	0.11
ESLR	62.20	-0.01	-0.49	0.01
LMT	21.77	0.13	0.6%	0.01
CD	26.6	-0.35	-1.3%	0.01
IOC	19.59	0.09	0.46%	0.11
TN	49.06	-0.16	-0.32%	0.01
TN	39.46	0.27	0.06%	0.01

FINANCIALS



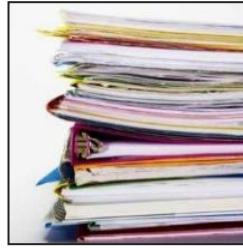
AVIATION
(VEHICLE INFO MGMT) (VEHICLE INFO MGMT)



RAIL & TRANSIT



ENERGY & UTILITY
MGMT



DOCUMENT MGMT



HUMAN
RESOURCE MGMT



PROJECT MGMT



QUALITY MGMT



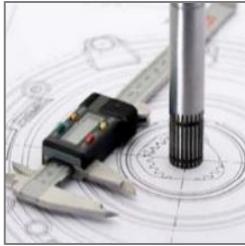
CALL CENTER
MGMT



BUSINESS
PERFORMANCE

Industrial Examples

A Car Manufacturer May Use



ENGINEERING



MANUFACTURING



MAINTENANCE



HUMAN
RESOURCE MGMT

SLB	-2.36	-0.23	-0.23
MAL	-2.42	-0.21	-0.21
NE	-2.44	-0.35	-0.35
MV	-2.45	-0.01	-0.01
OTVV	-1.13	-0.04	-0.04
HYOS	-21.14	-0.05	-0.05
PLUG	26.37	-0.04	-0.04
ESLR	62.20	-0.01	-0.01
LMT	21.77	0.52	0.86%
CD	26.6	0.12	0.5%
IOC	19.59	-0.35	-1.3%
TN	49.06	0.09	0.46%
	36.16	0.27	0.76%

FINANCIALS



PROJECT MGMT

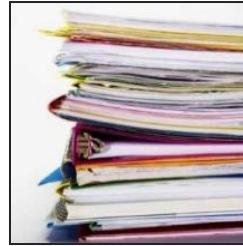


Industrial Examples

An Airline May Use



VEHICLE INFO MGMT



DOCUMENT MGMT



MAINTENANCE



HUMAN RESOURCE MGMT

FINANCIALS



TRAVEL MGMT



Industrial Examples

A Satellite TV Service Provider May Use



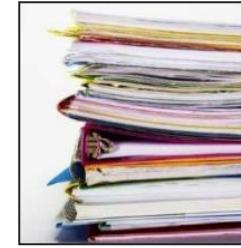
SERVICE MGMT



CUSTOMER RELATIONSHIP
MGMT (CRM)



BUSINESS
PERFORMANCE



DOCUMENT MGMT



Enterprise Application Vendors

Following are some of the leading Enterprise Application Vendors:



An Open Source BI



An Open Source BI



An Open Source CRM



An Open Source BI



An Open Source ERP



An Open Source ERP



An Open Source CRM



Extending Enterprise Applications to Mobile Devices

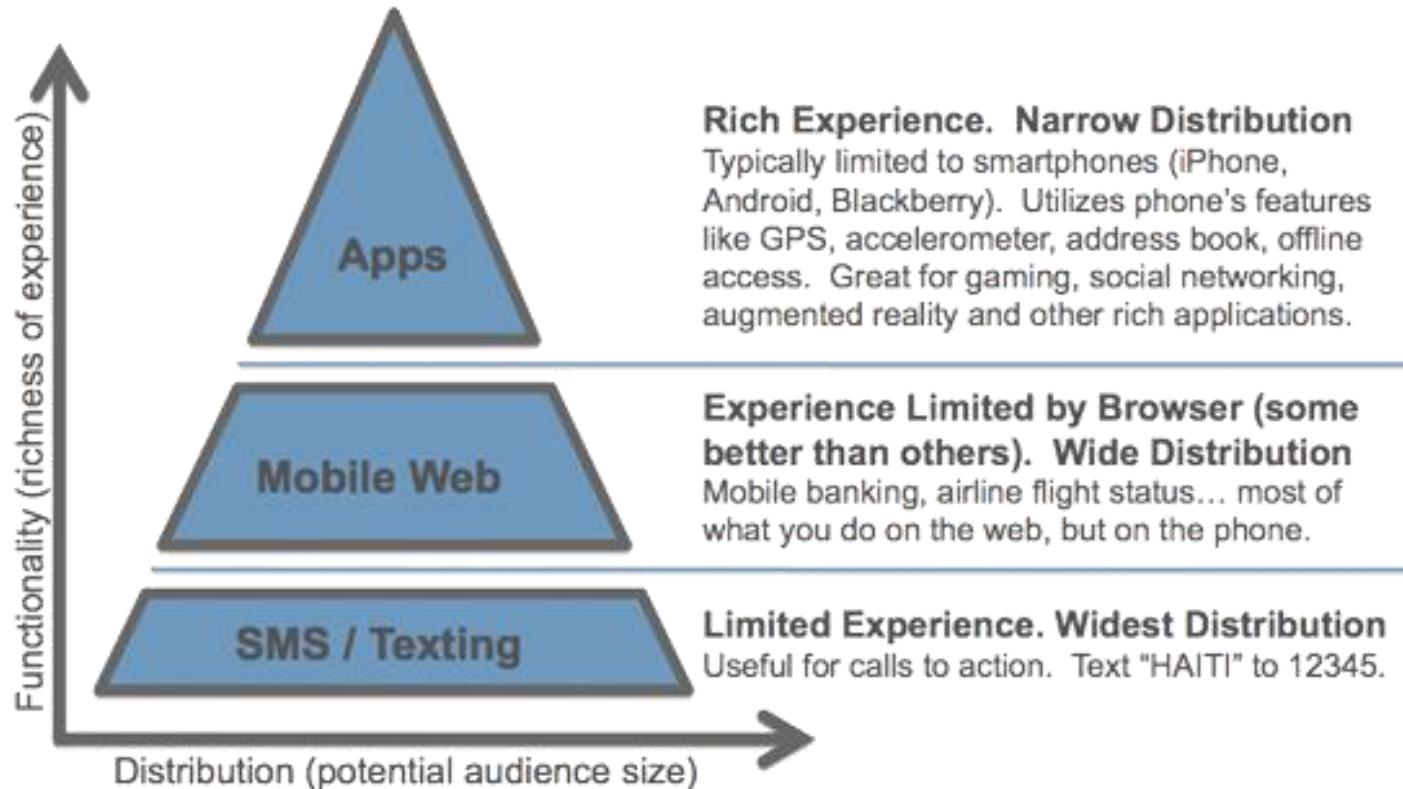
Why an EA need Mobile Applications?

- Some of the Enterprise Applications has Business Operations which are executed by their workforce on the field and on the go.
- Examples:
 - Maintenance
 - Manufacturing, Power Plant Machine Maintenance
 - Military Equipment Maintenance at Military Bases
 - Field Service Management
 - Satellite Television, Telecom Service Providers
 - Customer Relationship Management (CRM)
 - Courier Services (DHL, UPS etc.)
 - Distribution Management
 - Distribution Channels (Medicine, Milk Powder, Biscuits etc.)
 - Other
 - Offshore Drilling (Oil and gas)

Different Types Of Mobile Applications

Distribution of Apps

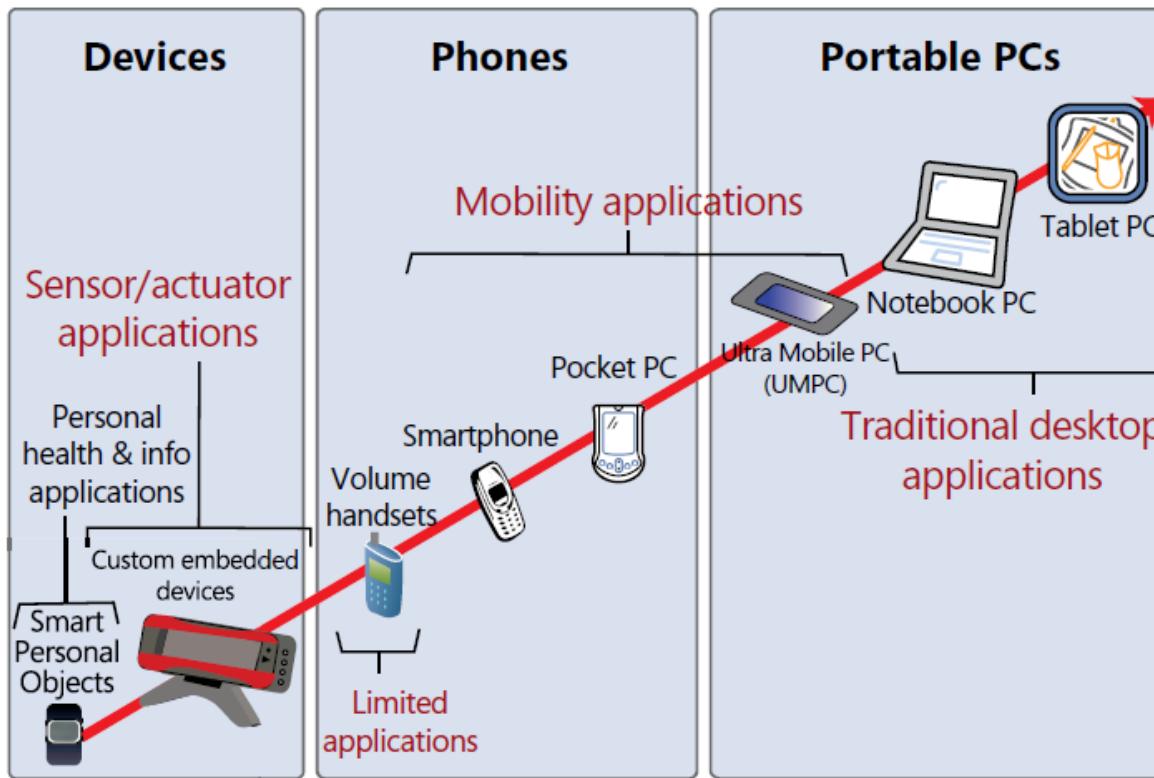
There are mainly three different types of Mobile Applications available today:



Different Types of Mobile Devices

The Range of Devices

There is a wide range of mobile devices available today:



Source: The Architecture Journal, Journal 14

Mobile Devices for Enterprise Apps

Rugged Devices to More Smart Phones



Tablet Computers



Windows Mobile Devices



Smart Phones
(iOS/Android/Windows)



Android Mobile Devices

Rugged Devices



Barcode Scanners



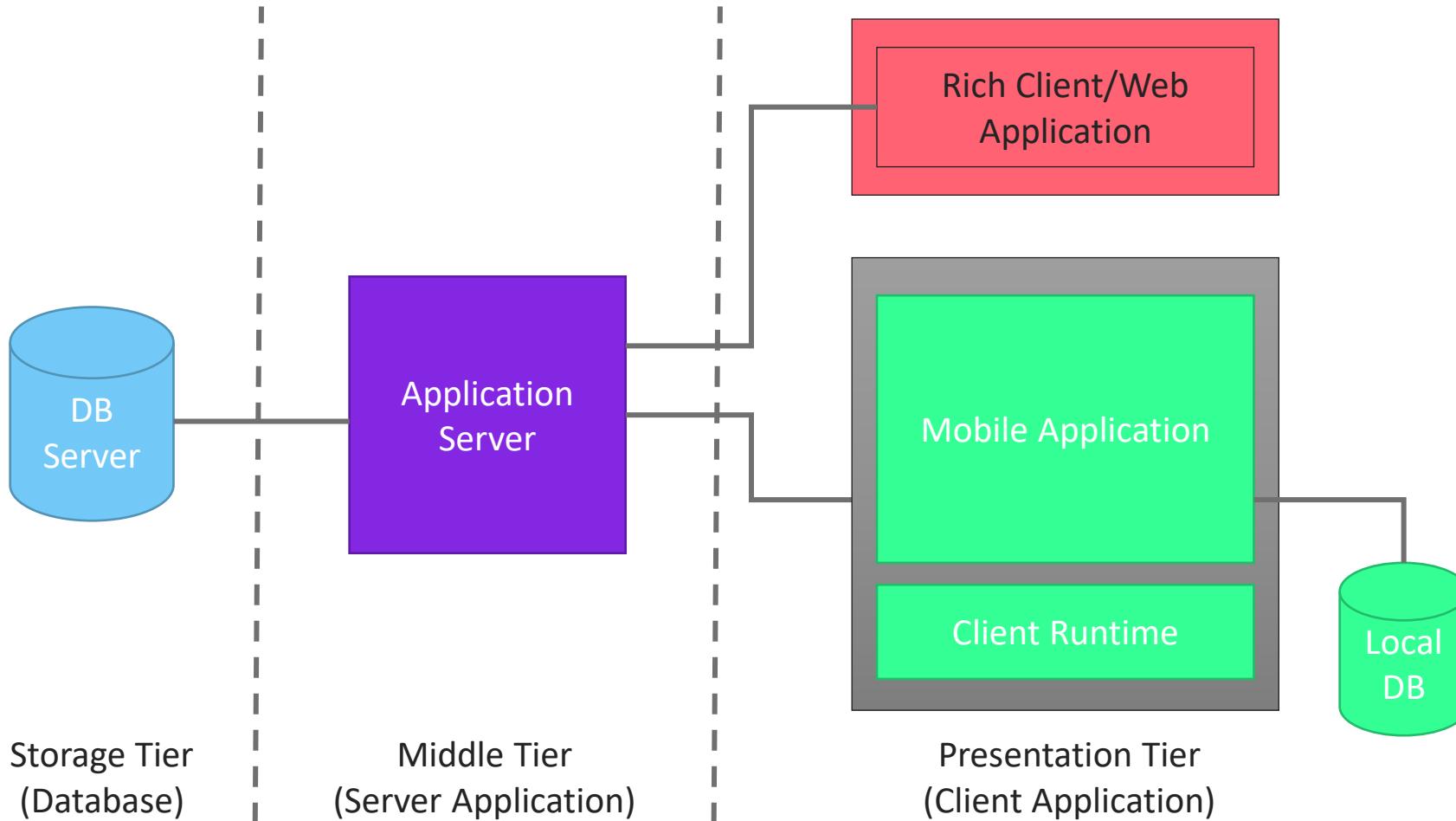
Mobile Printers

Extra Features

System Architecture

System Architecture

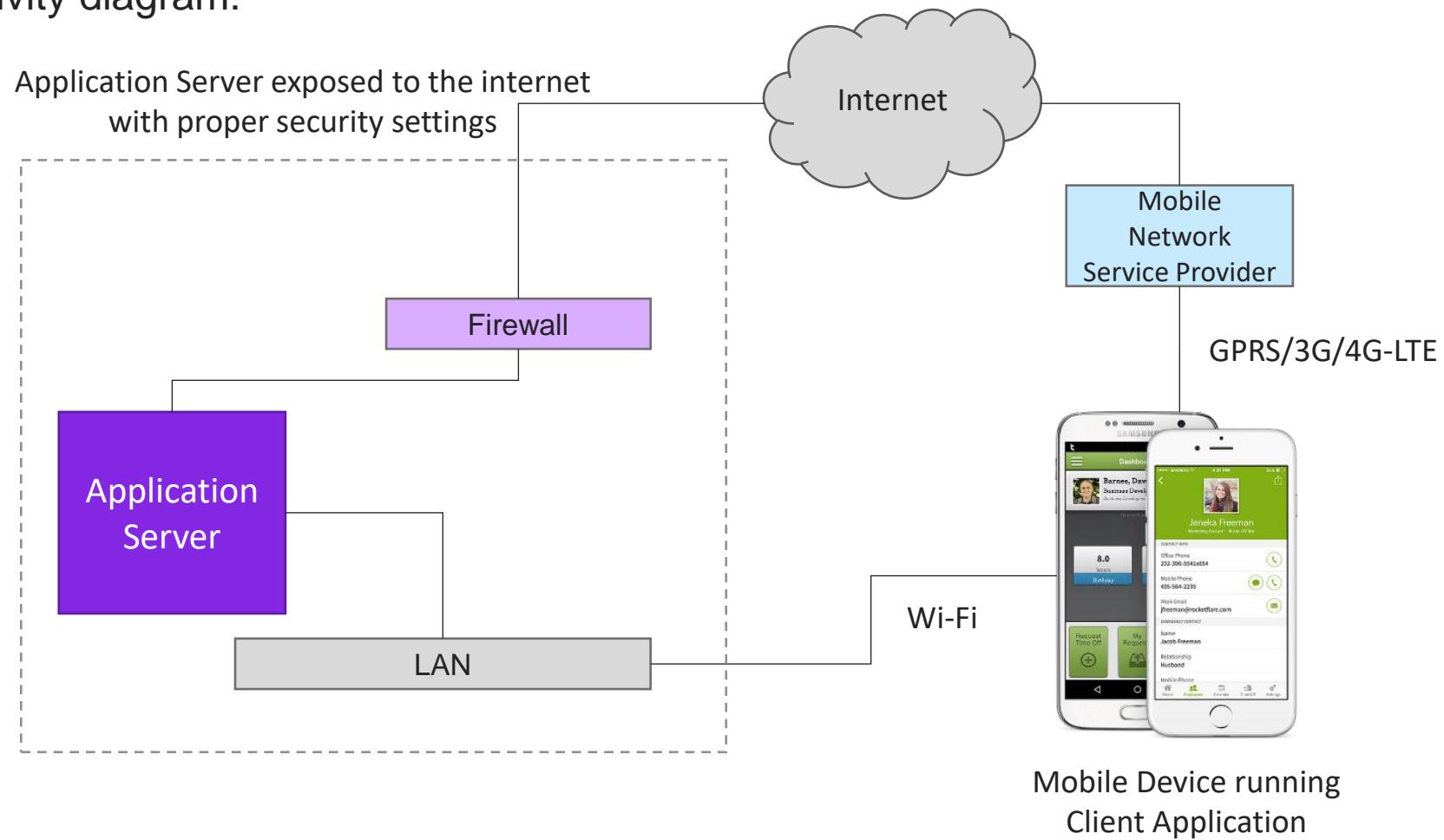
A Generic View



Connectivity

LAN(Wi-Fi)/GPRS/3G/4G-LTE

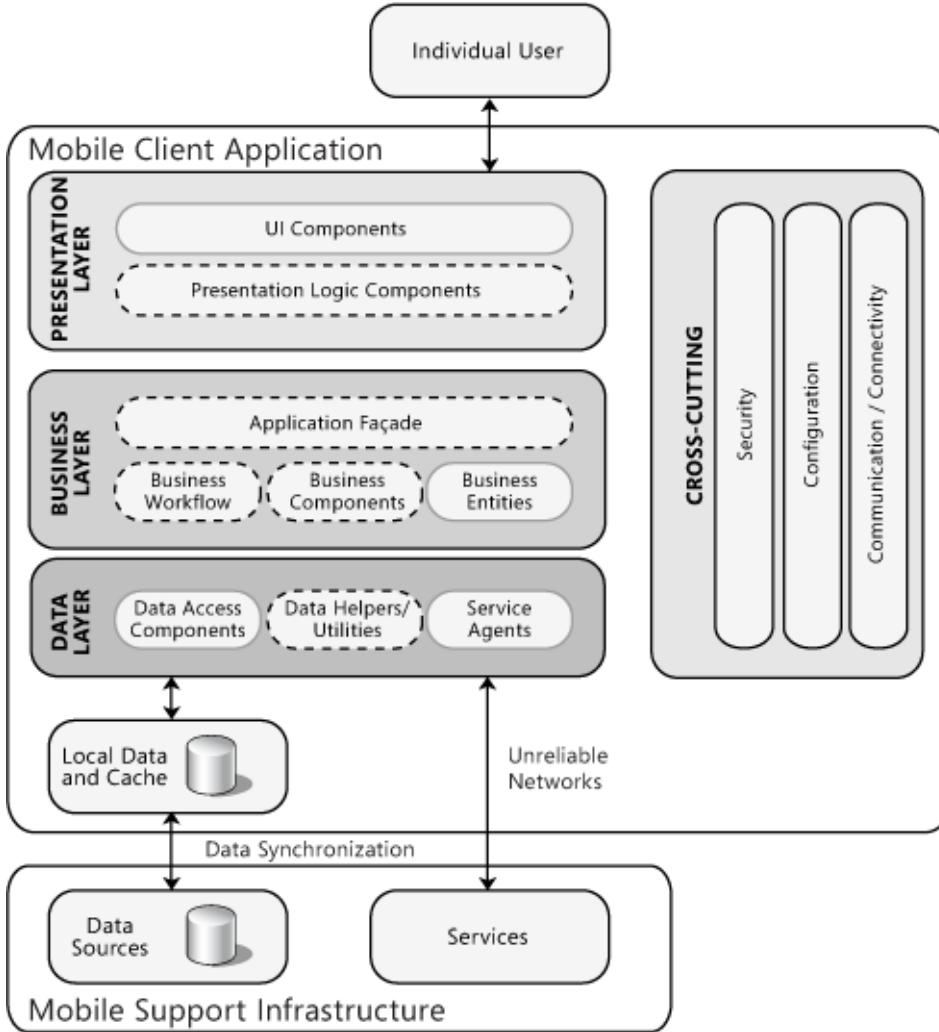
A sample connectivity diagram:



Application Architecture & Design

Mobile Application Architecture

A Typical Structure

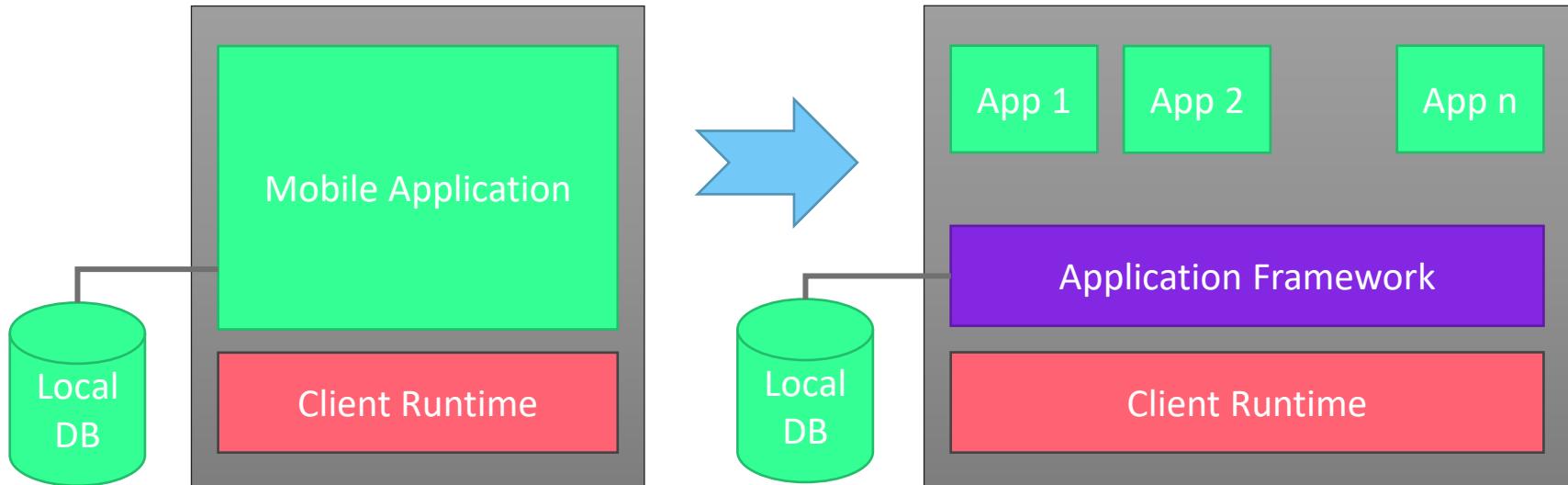


- A typical mobile application will be structured using:
 - Presentation Layer
 - Business Logic Layer
 - Data Layer
- Features like security, configuration and communication should be available for all layers.

Mobile Application Design

Extract Generic Features into a Separate Layer

- Extract all generic features of the application(s) and implement a framework.
- Write minimum amount of code in applications.

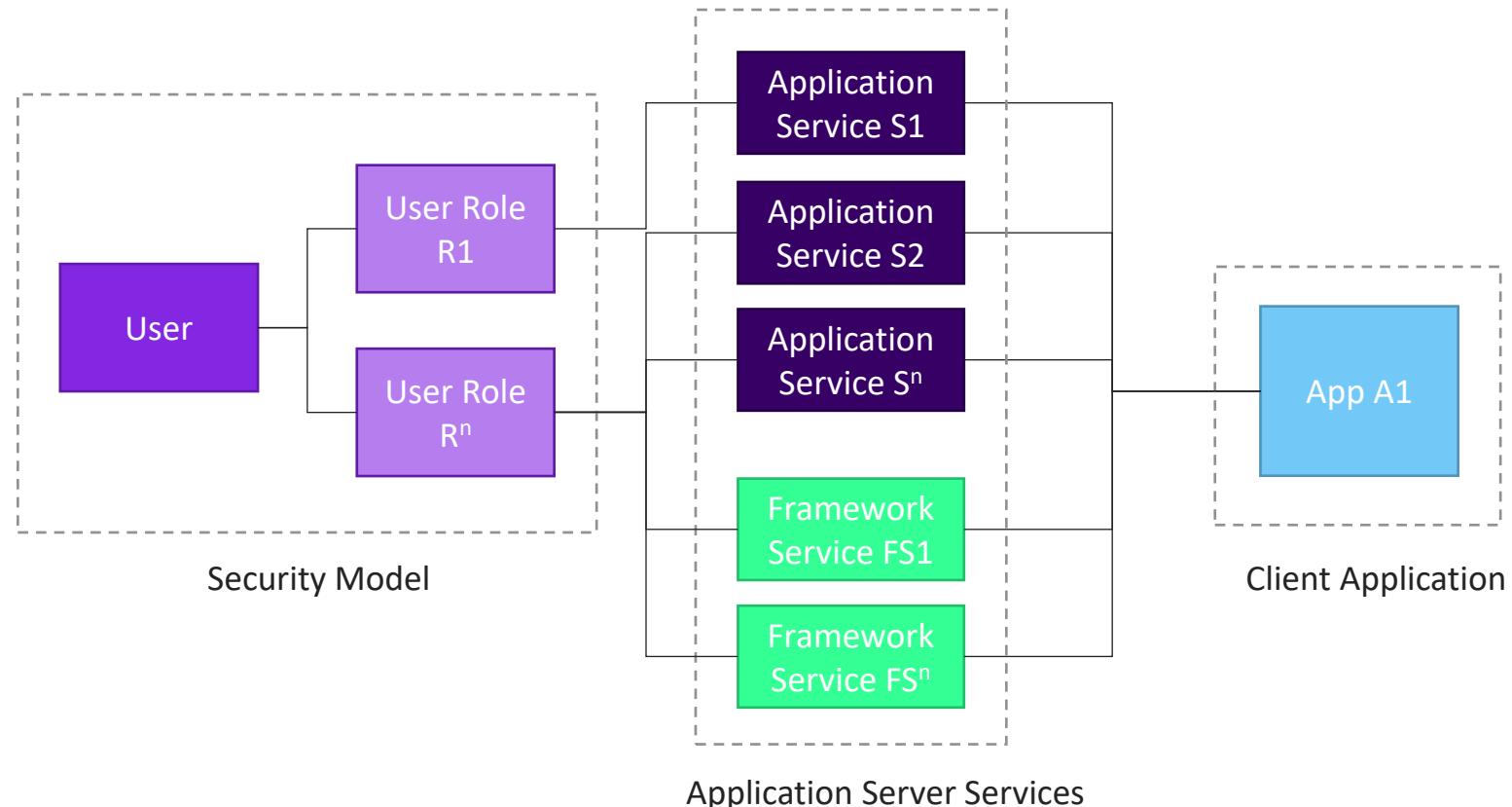


Application Model

Application Model

Design Model

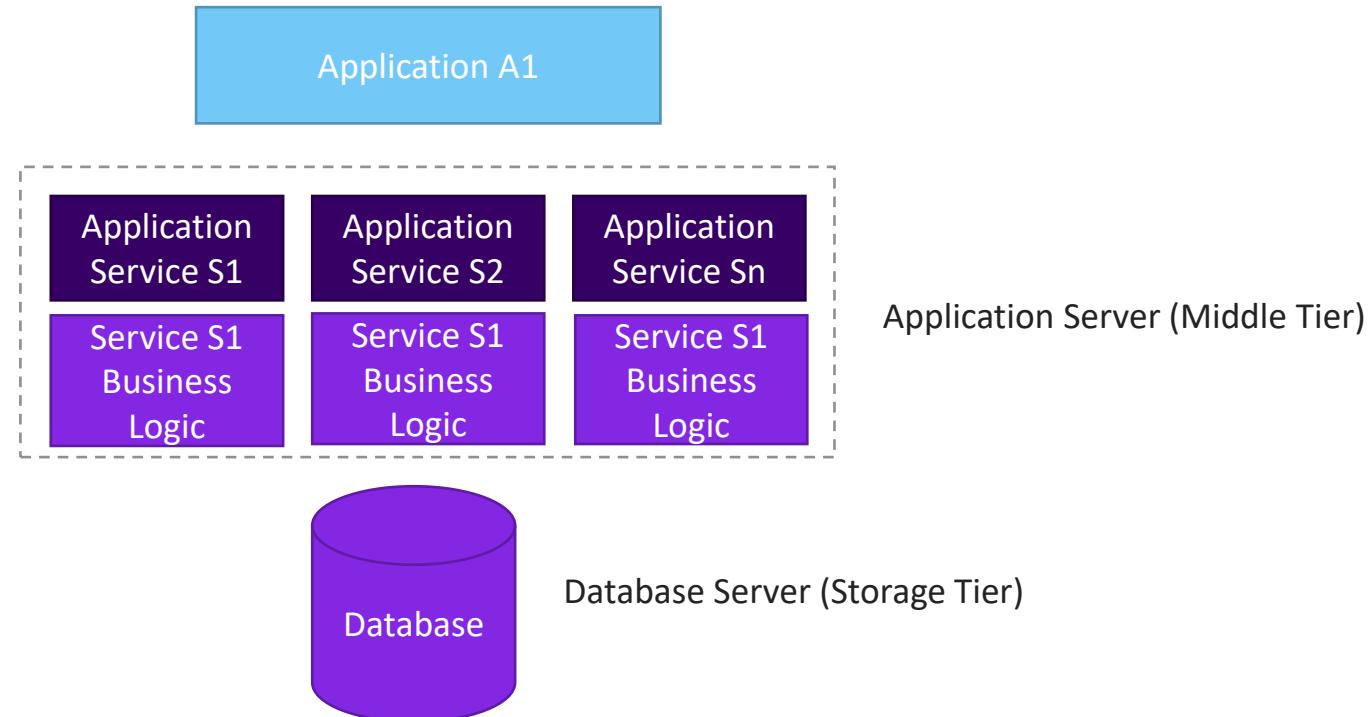
A mobile application will make use of set of applications and framework services in the server for communication.



Application Model

Business Logic Implementation

Business Logic for each Application Service either can be implemented in the application server or in the database.



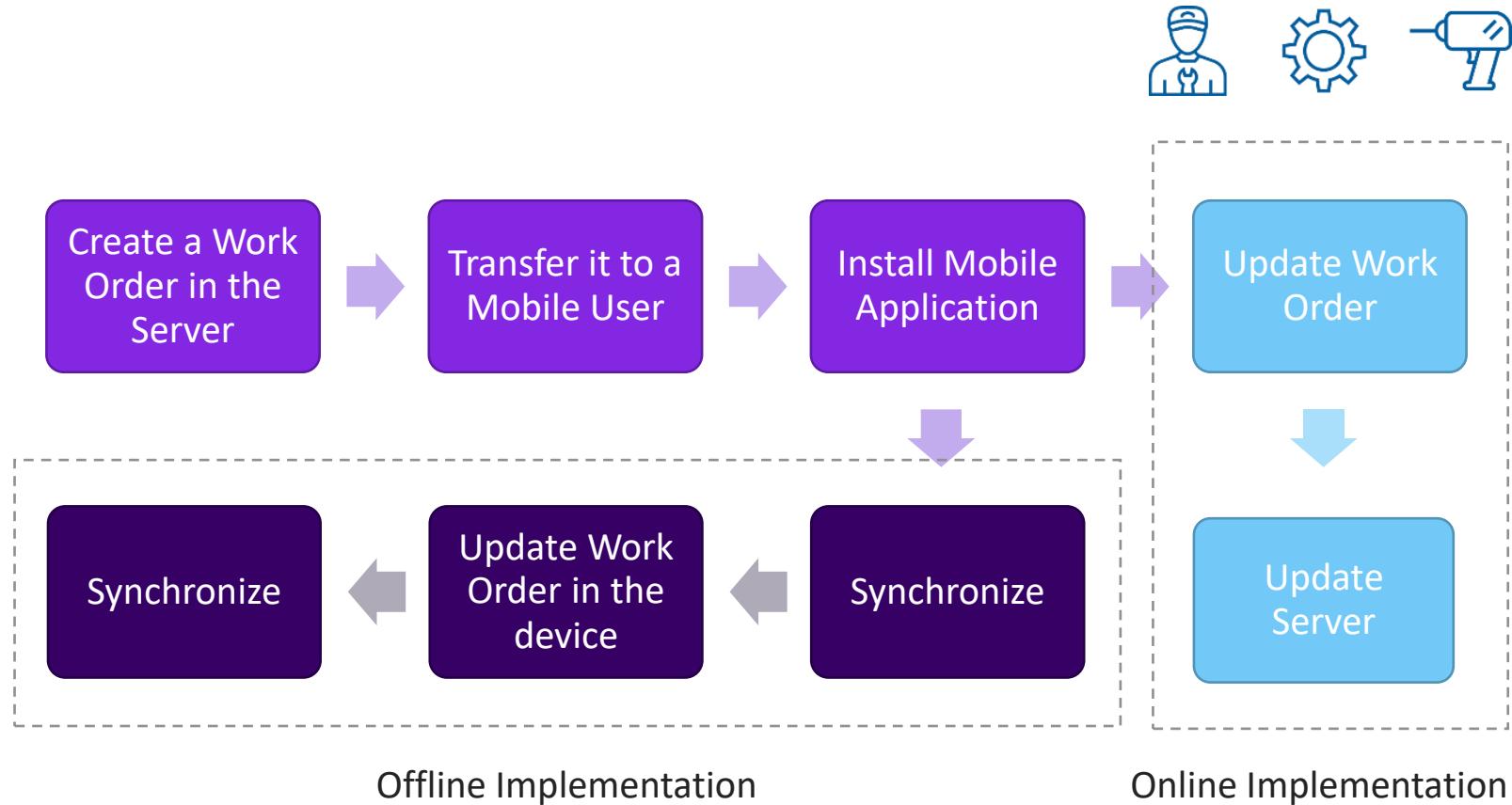
Application Model

Mobile Application Types Based On Connectivity

- Mobile Applications might need to be implemented as:
 - Online Applications
 - Only works with an online network connection.
 - No local database is used, applications directly communicate with the application server.
 - Offline Applications
 - Install apps and synchronize data to the device local database with an online connection.
 - Applications can be used even without a network connection.
 - Applications writes to the local database and modifications are then synchronized to the server.
 - Synchronization is normally two-way.

Application Model

A Sample Business Process For Field Service

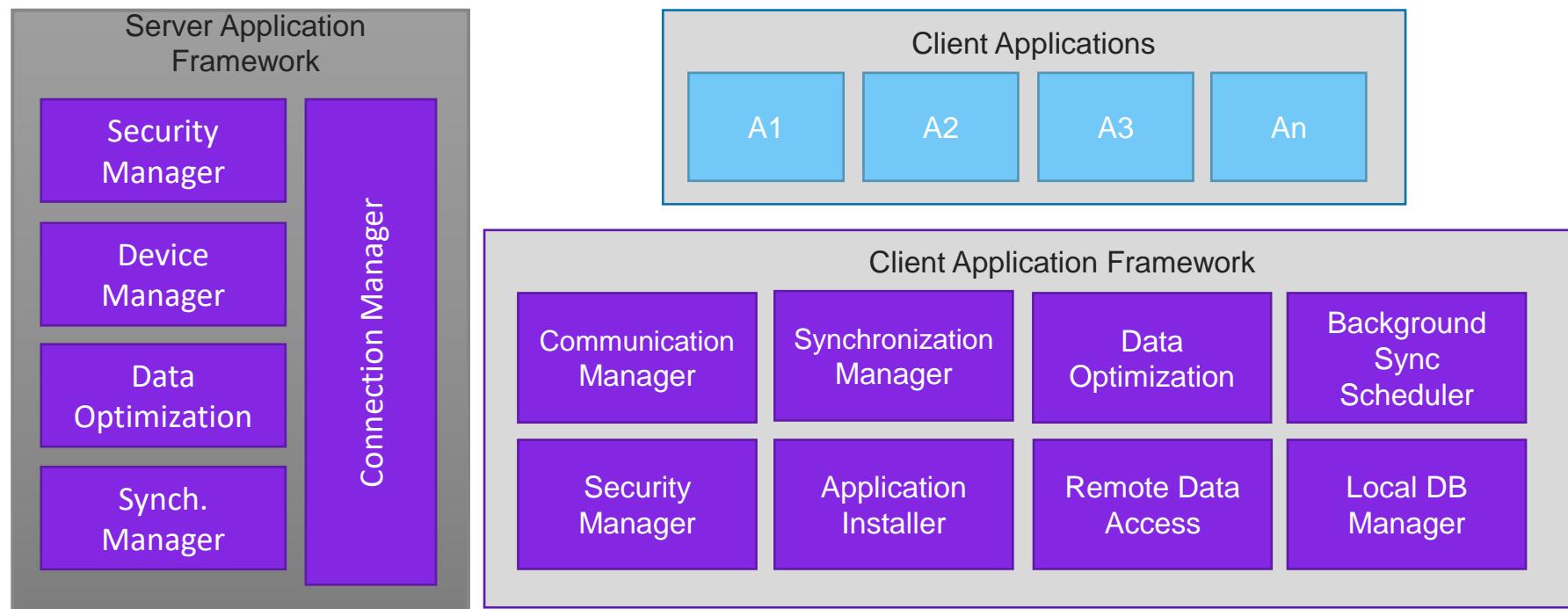


Application Framework Components

Application Framework Components

A Proof Of Concept

Application Framework will include features common for all mobile applications.



Note: A real life mobile application framework may contain more components.

Communication Manager

Client/ Server Communication



- Communication Manager is the central point of the client application which communicates with the server.
- All other framework components will invoke communication manager API methods to communicate with the server.
- It will use standard protocols for the specific Enterprise Application for sending and receiving messages between the client apps and the remote application server.
- Security Manager component will be used for User Authentication.

Synchronization Manager

Synchronizing Data to the Client



- Synchronization Manager component will be used by Offline Mobile Applications.
- It will invoke application's services to synchronize entity data to the client.
- Local DB Manager component will be used to create/update tables in the local database.
- Any changes that are done on the mobile app whilst offline are sent to the server.
- Synchronization can happen as push messages, a background task run on a schedule (Sync Scheduler), or on demand.
- Security Manager will be used for User Authentication.

Background Synch Scheduler

Scheduled Synchronization



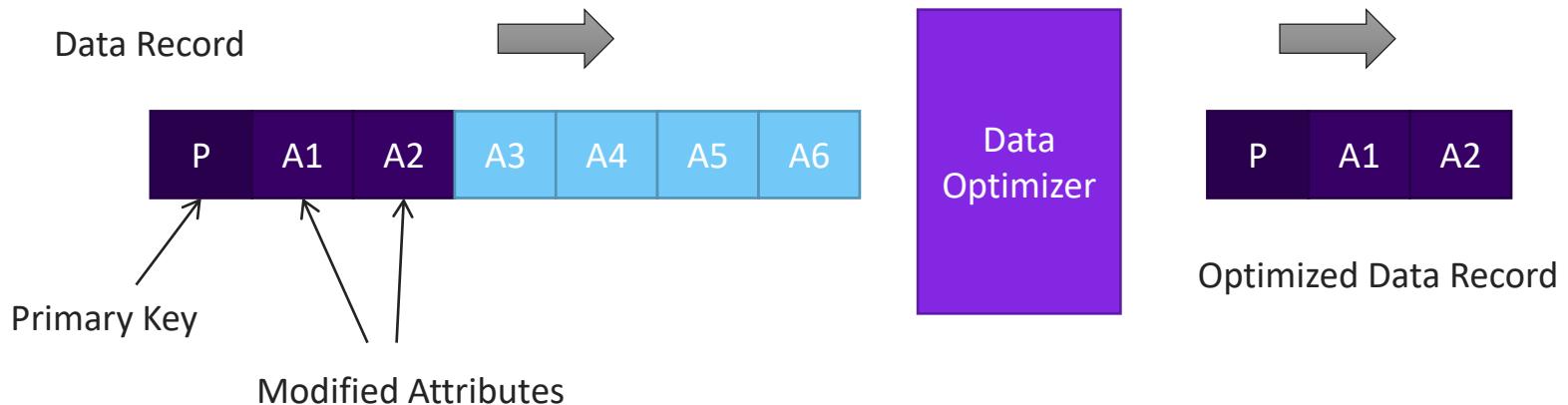
- Background Synchronization Scheduler component allows users to create scheduled synchronization processes as required.
- This module can be implemented to start a synchronization process at a given time or run it periodically after a given time.

Data Optimization

Reduce Network Overhead



- Data Transferred from the server and the client will be optimized to reduce the network overhead.
- The Optimizer will only include modified attributes (delta) of a given record when transferring on the network.
- It will also use compression algorithms such as GZIP.
- Data may be sent as JSON/XML and may use an encoding such as Base64.



Security Manager

User Authentication



- Security Manager will authenticate all server communication calls made by framework components via the Communication Manager.
- In addition, Application installation process will check security permissions for the required application server services.
- The local database on the device may also be encrypted to protect sensitive data.
- Local authentication mechanisms may employ passwords, PIN codes and biometric methods (fingerprint, iris scan).

Application Installer

Download Applications & Synchronize



- The database structure may change in the application server and the client apps need to reflect those changes.
- Application Installer will download application binary files from the server and synchronize metadata to the client (Offline Apps) to match the server.
- It will make use of the Local DB Manager to create and modify tables.
- Security Manager will be used to check security permissions granted for the current user.
- A metadata-driven approach reduces the number of times the user has to download and install a new version of the app.

Remote Data Access

For Online Applications



- Mobile Applications implemented in Online Mode will make use of the Remote Data Access component to access services on the server.
- This process will be managed by the Application Framework transparent to the applications.
- It will make use of data formats such as JSON/XML and Restful APIs.
- Security Manager component will be used for User Authentication.

Local Database Manager

CRUD Operations



- Local Database Manager component will be used to create tables in the local database and store synchronized data in the client.
- Application Installer will communicate with the Local Database Manager to execute this process.
- A standard SQL-based database for mobile applications is SQLite. However, a NoSQL approach using JSON files can also be used depending on the requirement.

Technology Stacks Available

Technology Stacks Available

Mobile Applications for Rugged Devices

- Following is the technology stacks available today for developing Enterprise Mobile Applications to run on rugged devices:

Local DB:
SQLite



Android Mobile Device
with Java Runtime

Technology Stacks Available

Mobile Applications for Smartphones

The following technology stacks can be used to develop enterprise applications for users who use smartphones:

Local DB:

SQLite, HanDBase etc.

Local DB:

SQLite, HanDBase etc...



Swift



iOS

iPhone Mobile Device



android

Android Mobile Device
with Java Runtime

Technology Stacks Available

Cross Platform Mobile Applications

- Using cross platform technology stacks means that a single code base can be used over multiple platforms.
- There are cross platform technologies that lets you create web-based or hybrid mobile applications.
- React Native, Ionic, Apache Cordova are some of the popular web-based framework which requires HTML5, CSS3 and JavaScript knowledge to make apps.
- Xamarin, on the other hand, is a hybrid framework. It will let you write apps in C# using Visual Studio and the end result is a platform-specific binary.



Thank you!

#MOMENTOFSERVICE



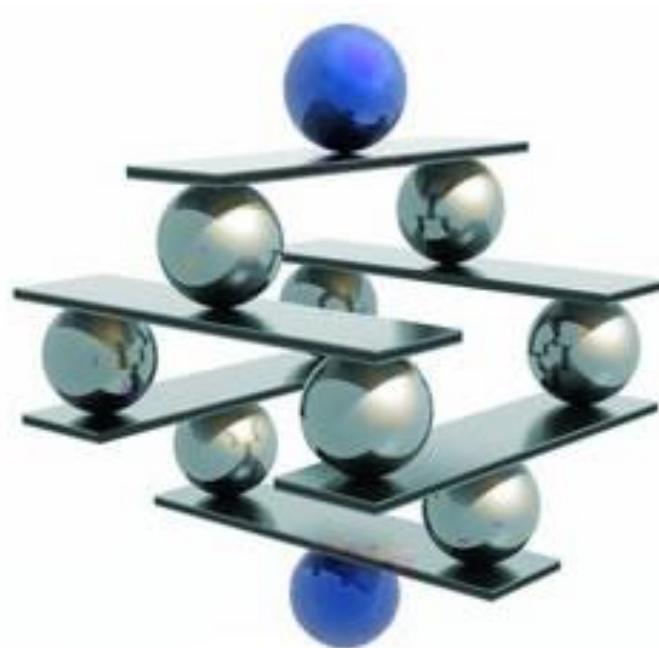


© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.

Introduction to SOAP, WSDL & Web Services

RESTFULL Web services, WEB API, SSL and OpenID



Why we need web services

- Exposing the existing function on to network:
- Connecting Different Applications (Interoperability)
- Standardized Protocol



Characteristics of Web services

XML-Based

- Web Services uses XML at data representation
- highly interoperable application at their core level.

Loosely Coupled

- A consumer of a web service is not tied to that web service directly.
- A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated.
- software systems more manageable and allows simpler integration between different systems.

Coarse-Grained

- Object-oriented technologies such as Java expose their services through individual methods.
- Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.
- Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

Ability to be Synchronous or Asynchronous

- In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing.
- Asynchronous operations allow a client to invoke a service and then execute other functions.

Supports Remote Procedure Calls(RPCs)

- Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.
- A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

Supports Document Exchange

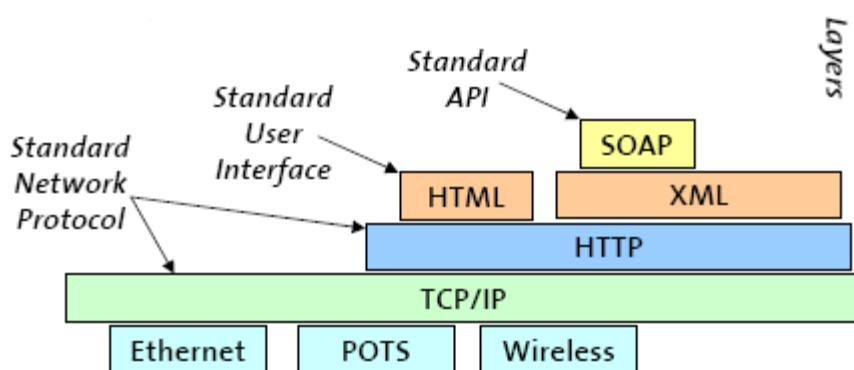
- One of the key advantages of XML is its generic way of representing not only data, but also complex documents.

What is SOAP

- RPC is an obvious and popular paradigm for implementing the client-server model of distributed computing.
- Different RPC protocols exist today.
 - Sun RPC, DCE/RPC, DCOM, CORBA
- Most of the RPC protocols are too low level and certainly not compatible among each other (gateways are needed)
- Since RPC represents a compatibility and security problems, firewalls and proxy servers will normally block this kind of traffic.
- To address this problem, XML was used to define SOAP.
- Soap uses HTTP/SMTP as the communication protocol

Why is SOAP

- XML-based protocol for exchanging information between applications
- Primary focus of SOAP is Remote Procedure Calls transported via HTTP
- Similar to DCOM, CORBA, and Java RMI, the main difference is that SOAP messages are written entirely in XML
- SOAP is therefore uniquely platform and language independent
- For example, a SOAP Java client running on Linux or a Perl client running on Solaris can connect to a Microsoft SOAP server running on Windows 2000

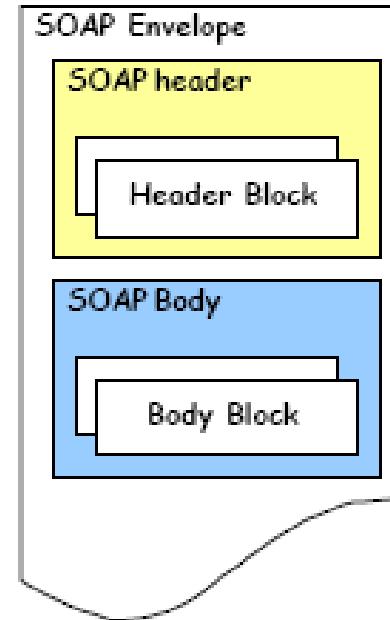


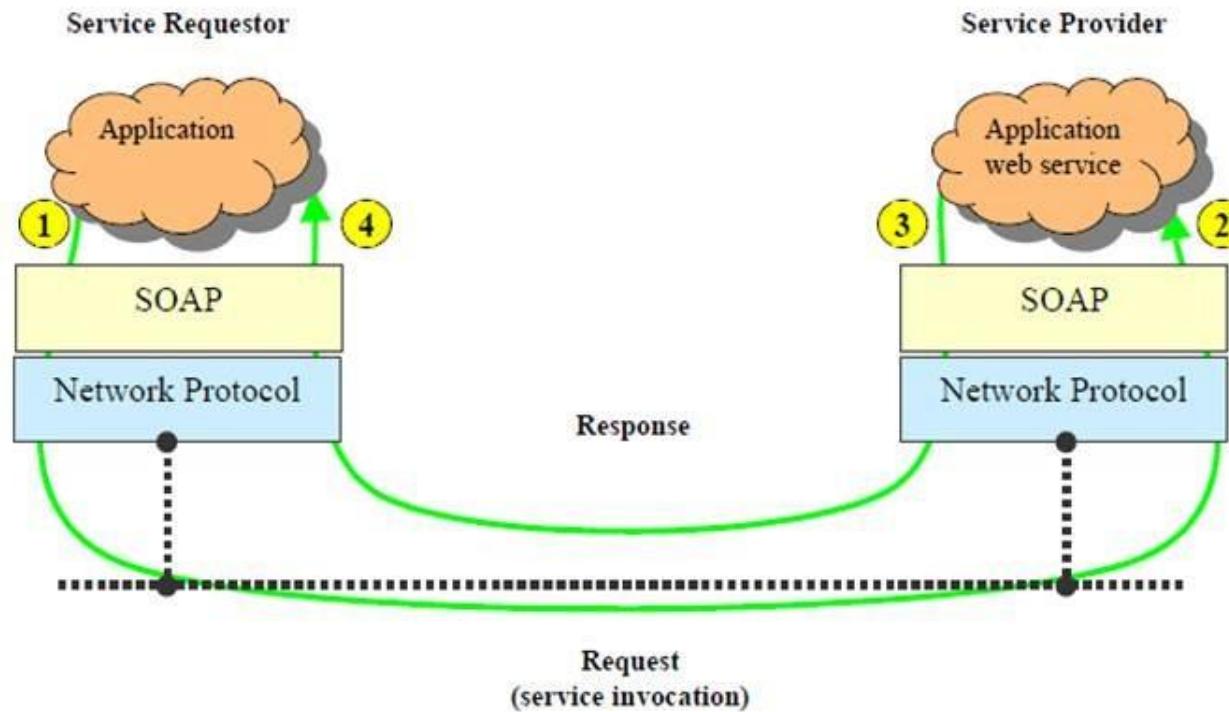
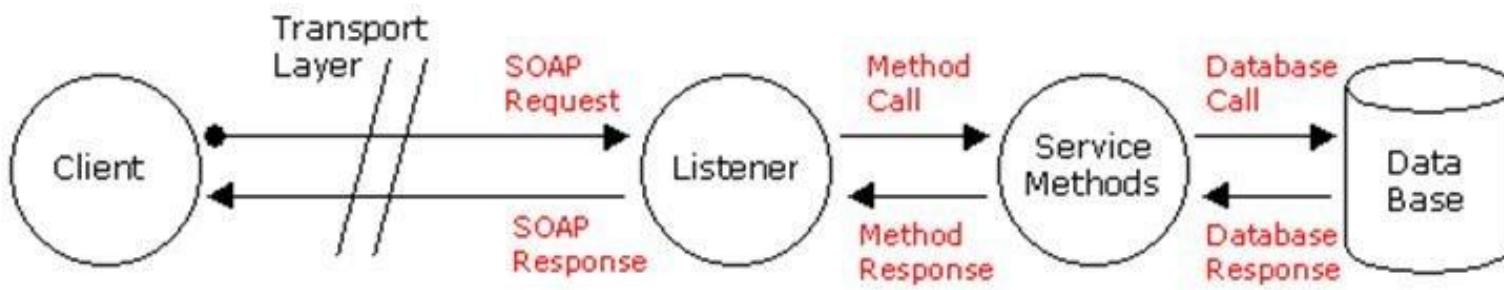
How SOAP works

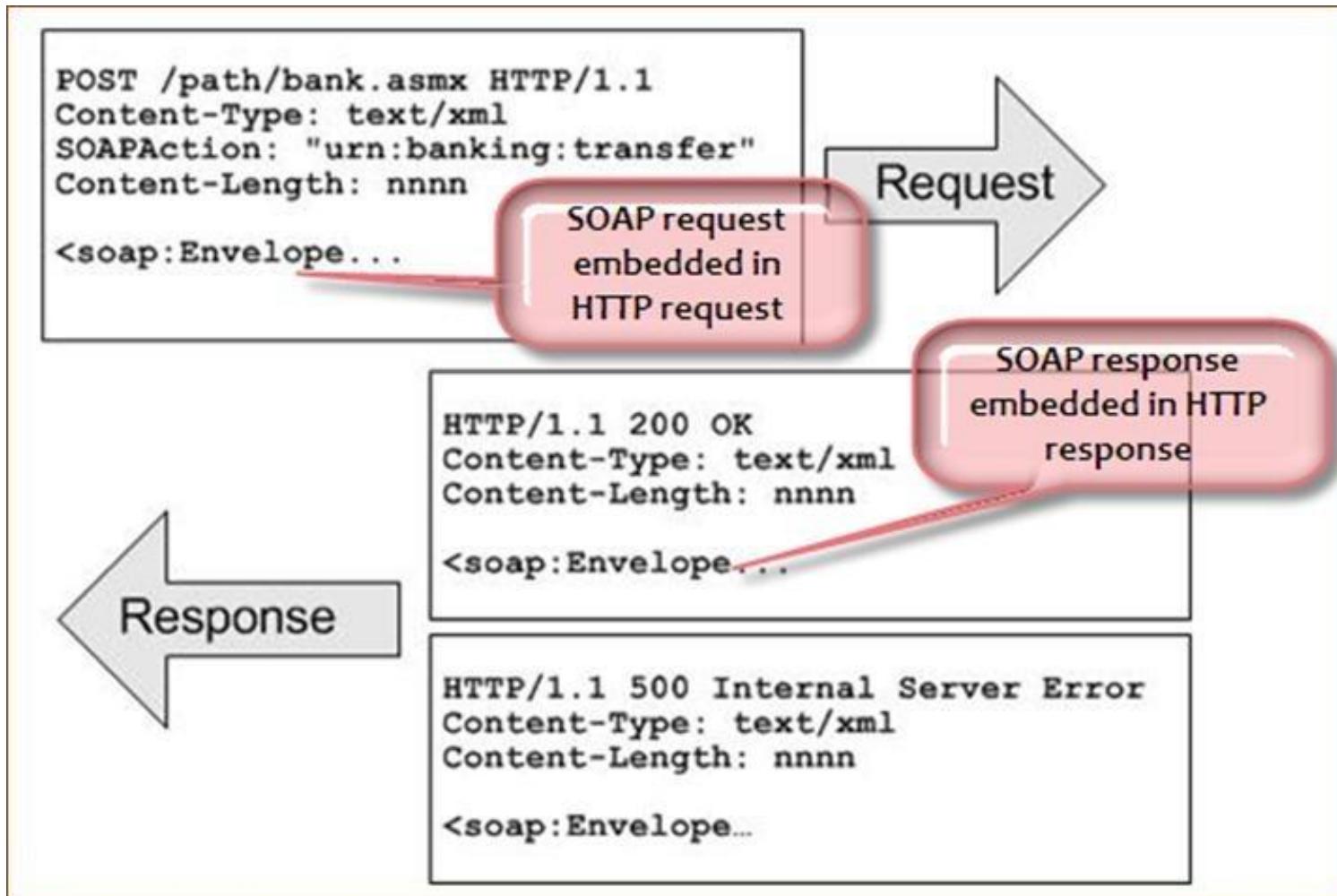
- RPC Using HTTP
 - At the Client
 - Turns the RPC call into a XML document
 - Transports the XML data over HTTP (or SMTP) (Client to Server)
 - At the Server
 - Turns the XML document into a procedure call
 - Turns the procedure's response into a XML document
 - Transports the XML data over HTTP (or SMTP) (Server to Client)
 - At the Client
 - Turns the XML data into the response to the RPC

Structure of a SOAP Message

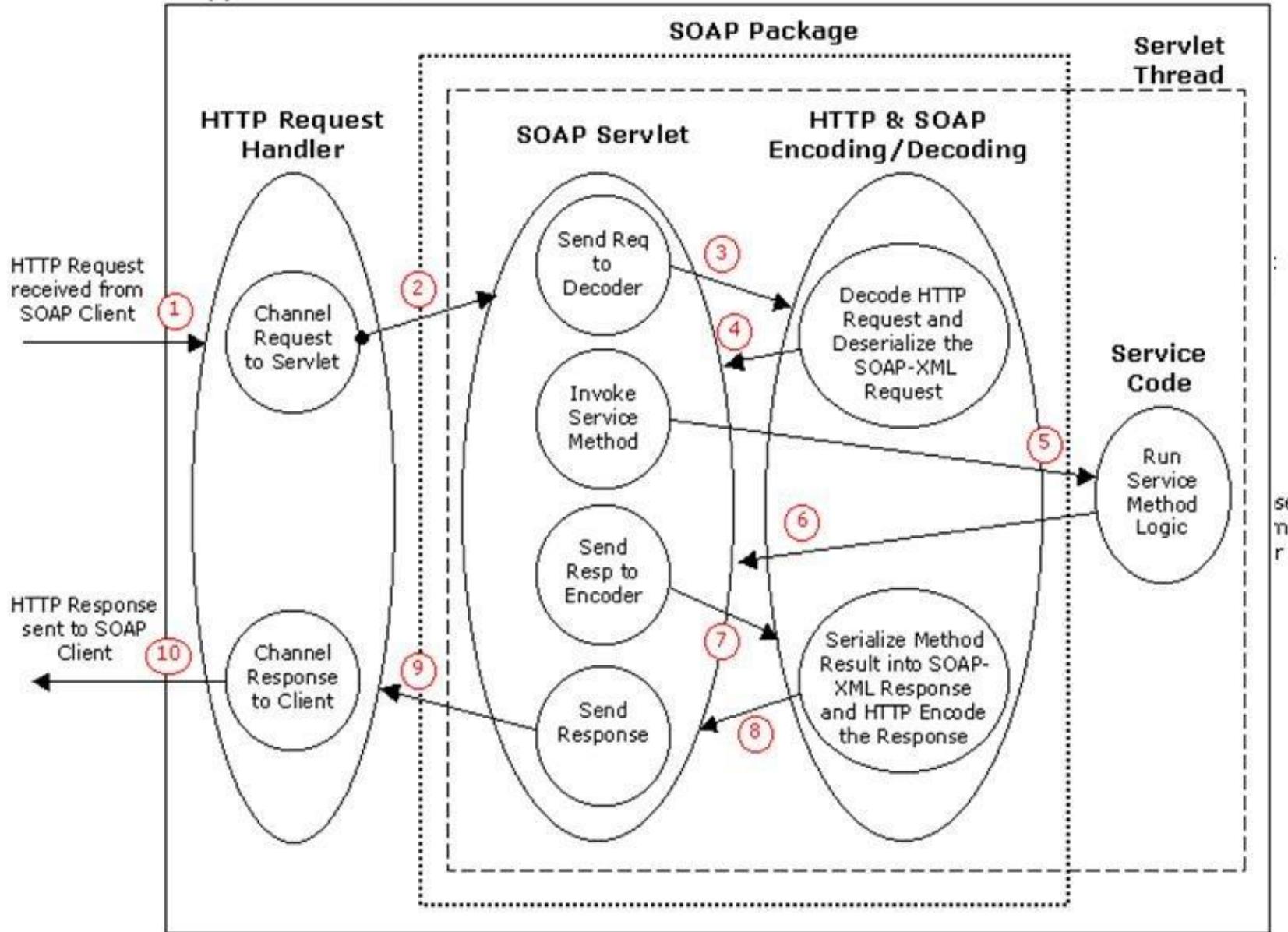
- SOAP is based on message exchanges.
- Messages are seen as envelopes where the application encloses the data to be sent.
- A message has two main parts; header and body, which both can be divided into blocks.
- SOAP does not specify what to do in the body, it only states that the header is optional and the body is mandatory.
- The use of header and body, however, is implicit. The body is for application level data. The header is for infrastructure level data.







Appserver Process

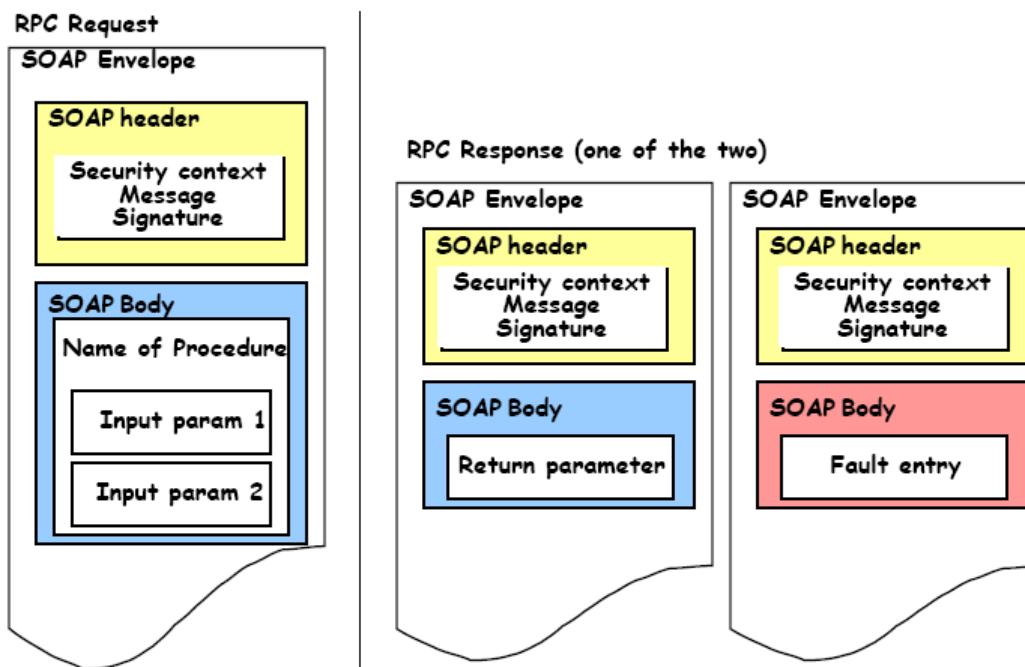


SOAP Header

- The header is intended as a generic place holder for information that is not necessarily application dependent (the application may not even be aware that a header was attached to the message).
- Typical uses of the header are: coordination information, identifiers (for, e.g.: transactions) and security information (e.g.: certificates)

SOAP Body

- The body is intended for the application specific data contained in the message.
- Also it may contain a Fault entry Section (for reporting errors in processing a SOAP message)



SOAP Message

XML name space identifier for SOAP serialization
XML name space identifier for SOAP envelope

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Header>
        <t:Transaction
            xmlns:t="some-URI"
            SOAP-ENV:mustUnderstand="1">
            5
        </t:Transaction>
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DEF</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Introduction to WSDL

```
<?xml version="1.0" encoding="I
<definitions name="AktienKurs":
    targetNamespace="http://loca ....
    xmlns:xsd="http://schemas.xmlsoap.or
    xmlns="http://schemas.xmlsoap.org/wsd
    <service name="AktienKurs">
        <port name="AktienSoapPort" binding
            <soap:address location="http://loc
        </port>
        <message name="Aktie.HoleWert">
            <part name="body" element="xsd:Tra
        </message>
        ...
    </service>
</definitions>
```

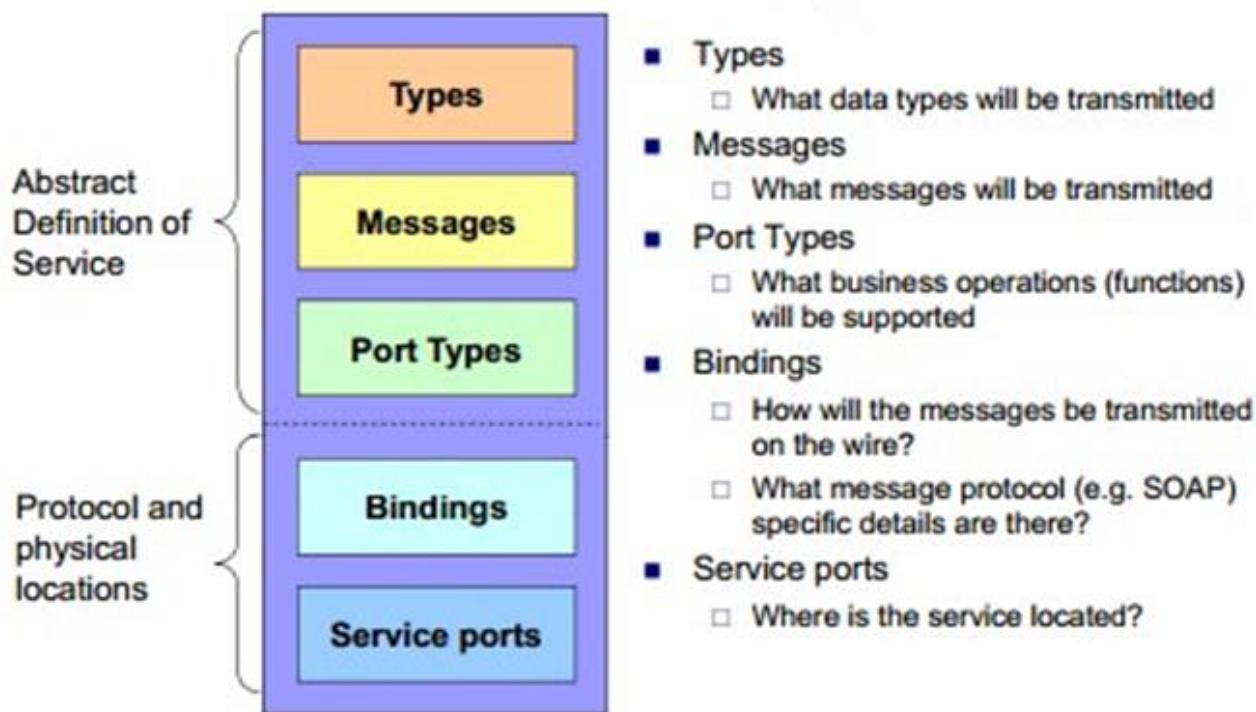
WSDL

What is WSDL?

- WSDL stands for Web Services Description Language
- WSDL is an XML document
- WSDL is used to describe Web services
 - It specifies the location of the service and the operations (or methods) the service exposes
- WSDL is also used to locate Web services

The WSDL Document Structure

- A WSDL document describes a web service using these major elements:



WSDL Types

- The **<types>** element defines the data types that are used by the web service.
- For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Messages

- The **<message>** element defines the data elements of an operation.

```
<message name="newTermValues">
    <part name="term" type="xs:string"/>
    <part name="value" type="xs:string"/>
</message>
```

- Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

WSDL Port

- A WSDL port describes the interfaces (legal operations) exposed by a web service.
- It defines a **web service**, the **operations** that can be performed, and the **messages** that are involved.
- Operation Types
 - One-way
 - Request-response
 - Solicit-response
 - Notification

One-Way Operation

```
<message name="newTermValues">
    <part name="term" type="xs:string"/>
    <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
    <operation name="setTerm">
        <input name="newTerm" message="newTermValues"/>
    </operation>
</portType >
```

Request-Response Operation

```
<message name="getTermRequest">
    <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
    <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
    <operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
    </operation>
</portType>
```

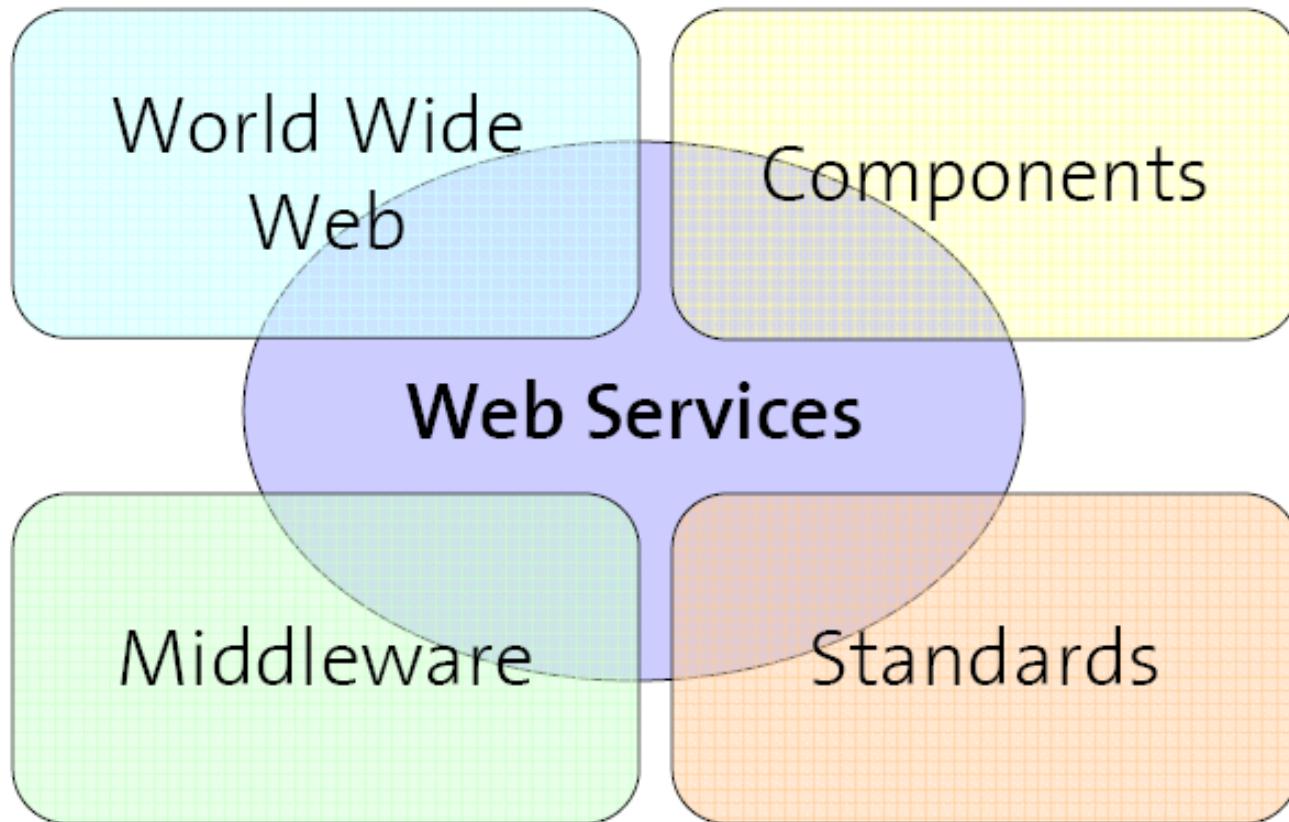
WSDL Binding

- WSDL bindings defines the message format and protocol details for a web service.
- The **binding** element has two attributes - name and type.
 - The name attribute defines the name of the binding, and the type attribute points to the port for the binding.
- The **soap:binding** element has two attributes - style and transport.
- The **operation** element defines each operation that the port exposes
 - For each operation the corresponding SOAP action has to be defined. You must also specify how the input and output are encoded. .

Introduction to Web Services



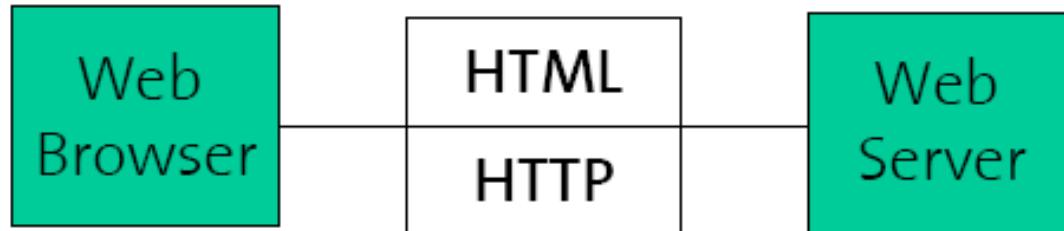
Web Services



Web-based Services & Web Services

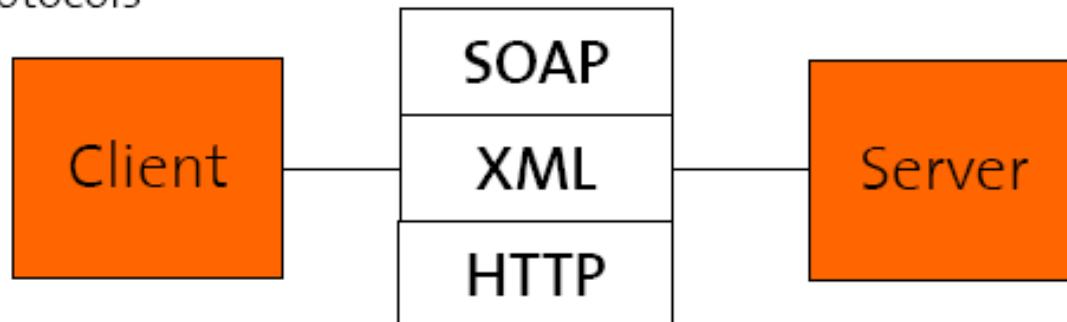
Web-based Services

- Services offered through a Web site



Web Services

- Services offered through Web-wide standardized protocols



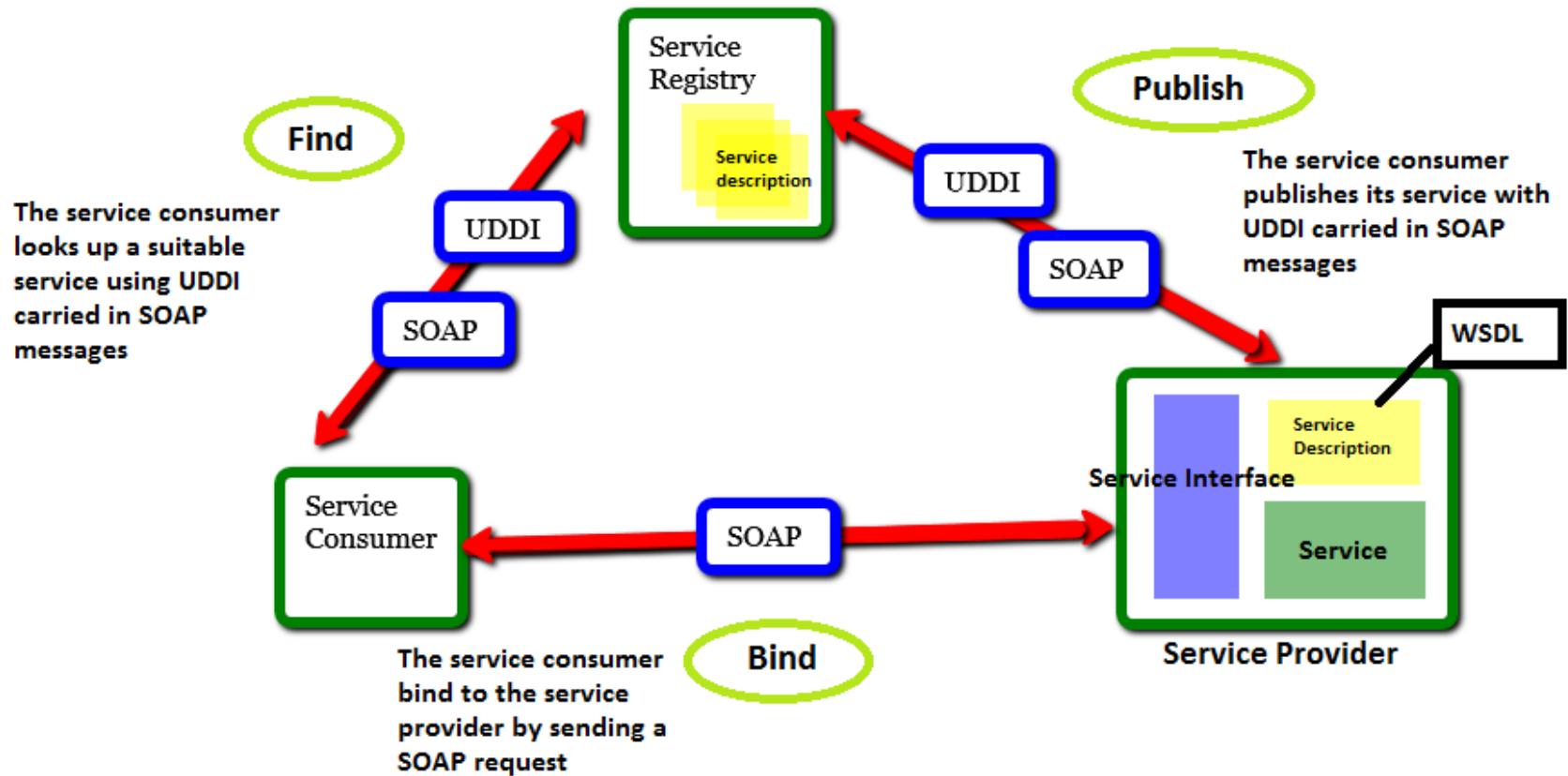
Web Service (W3C Definition)

- “A Web service is a software application **identified by a URI**, whose interfaces and bindings are capable of being **defined, described and discovered as XML artifacts**. A Web service supports direct interactions with other software agents using **XML-based messages** exchanged via the **Internet based protocols**”

Web Service

- Web Service definition by W3C emphasizes different aspects:
 - In order to be accessible, a service should be defined, described and discovered.
 - XML is the foundation for all standards that are going to be used (SOAP, WSDL, UDDI)
 - Web services are components that can be readily integrated into more complex distributed applications.
 - Web services are meant for software based consumption (while Web-based applications are meant to be used by humans equipped with a WWW browser)

Web Service Architecture



Web Service Roles

- Service Provider

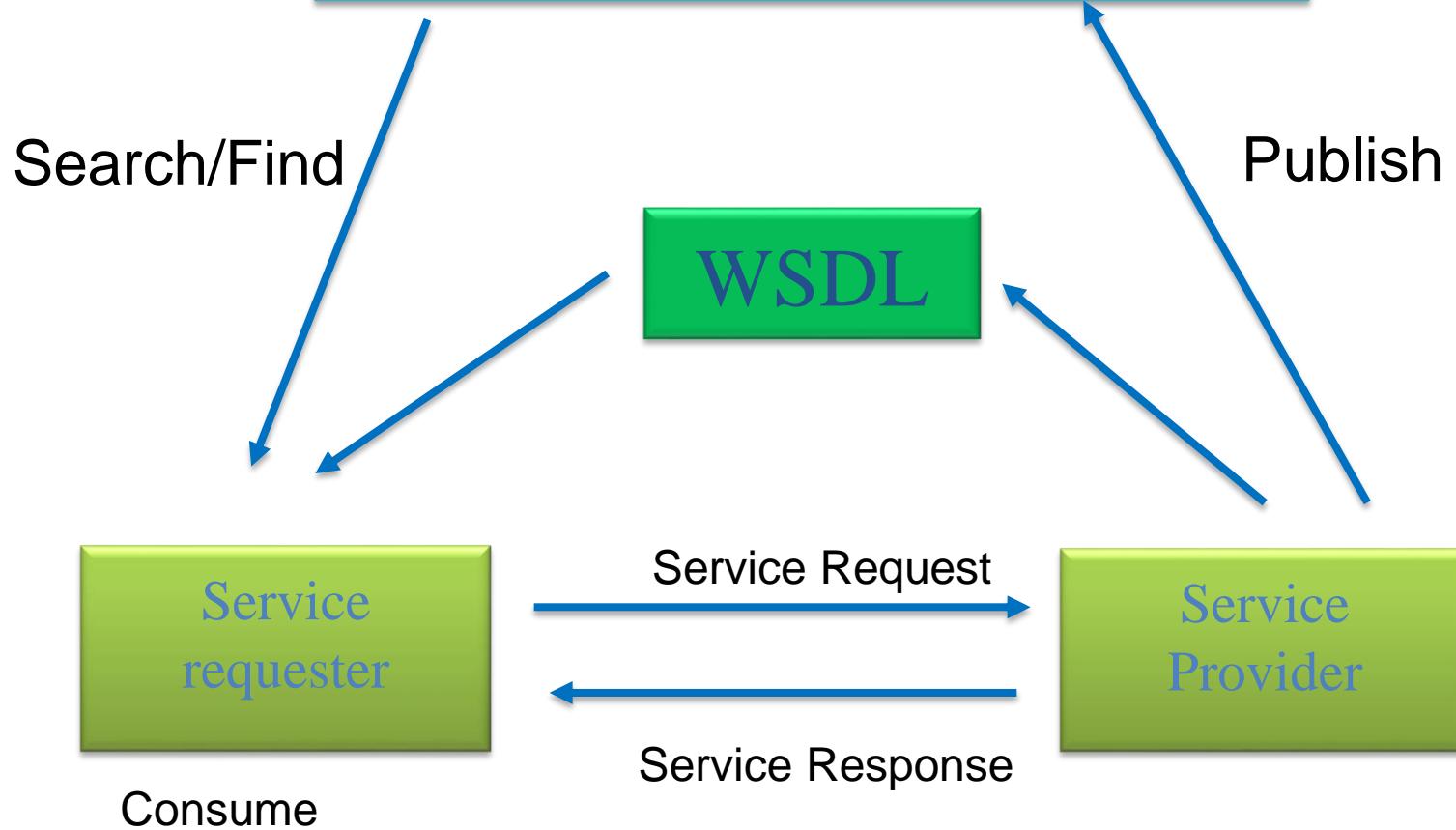
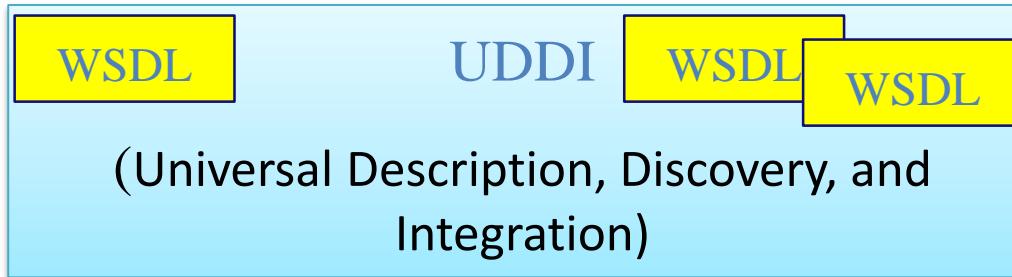
This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

- Service Requestor

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

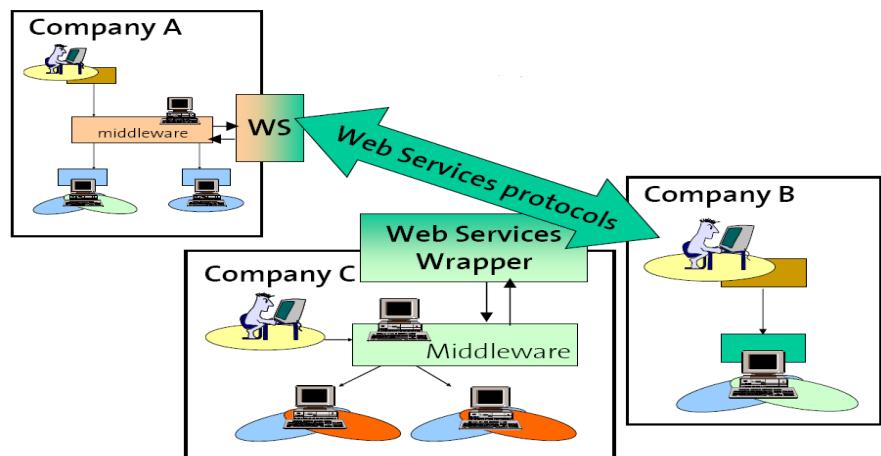
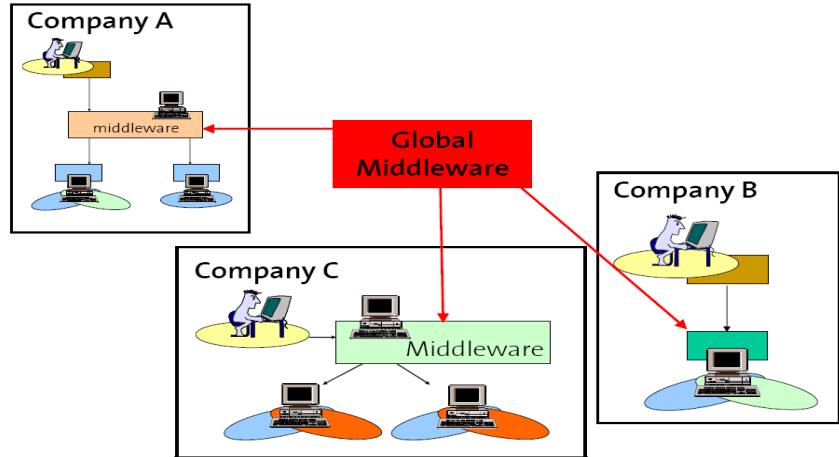
- Service Registry

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.



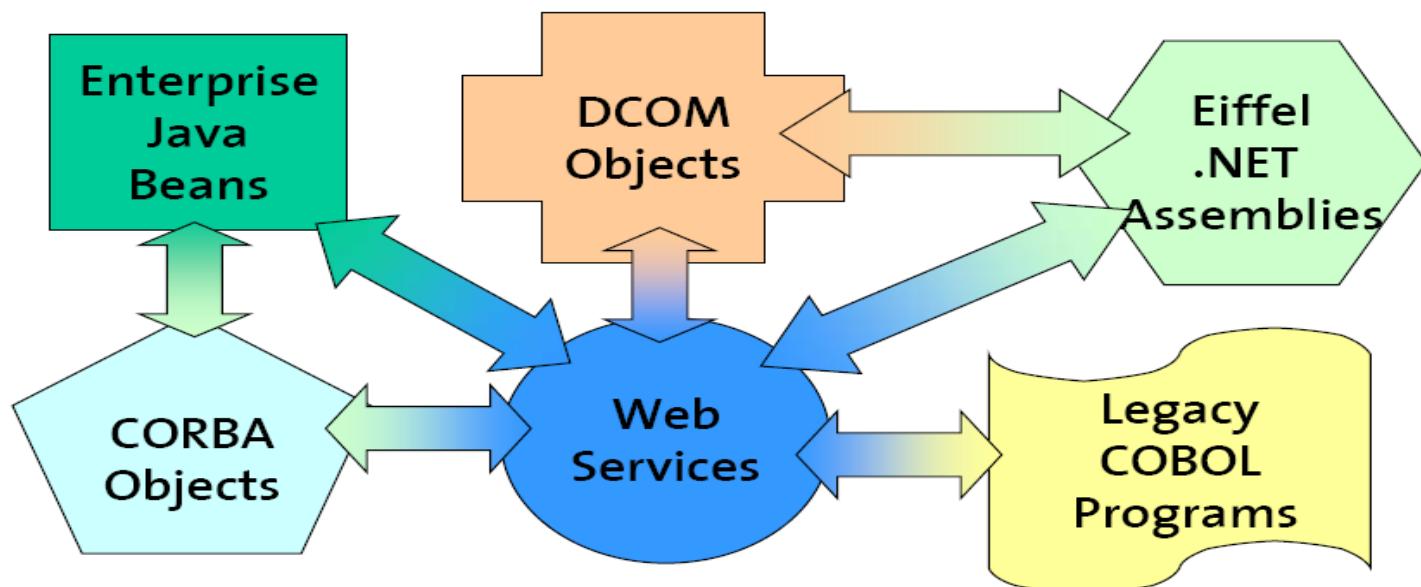
Benefits of Web Services

- The Web services architecture represented by SOAP, UDDI, and WSDL is a direct descendant of conventional middleware platforms
- They can be seen as the most basic extensions that are necessary to allow conventional synchronous (RPC-based) middleware to achieve interoperability



Benefits of Web Services

- Platform independence (Hardware, OS)
- Programming language neutrality
- Portability across Vendor/Middleware tools



Benefits of Web Services

- Low Cost Communication

Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

- Interoperability

Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services

- Standardized Protocol

Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.

What Is RESTFULL Web Service

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

the life cycle of a RESTful web service is **per request** so the service does not have to worry about concurrency and can use instance variables safely.

HTTP methods All client server communication on the World Wide Web are done using the following simple HTTP methods:

- **GET**= "give me some info" (Retrieve)
- **POST**= "here's some info to new Info" (Create)
- **PUT**= "here's info to update" (Update)
- **DELETE**= "delete some info" (Delete)

Two Type of Web Services

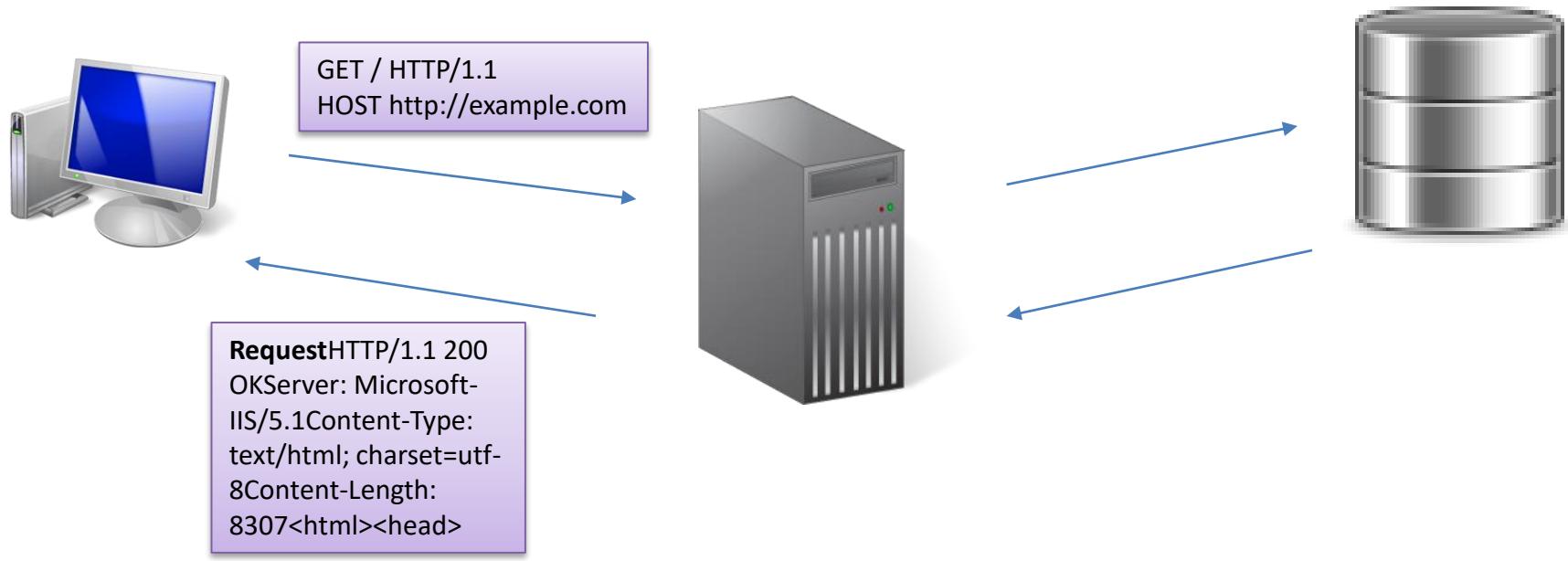
1. Service Oriented web services – mostly used in Enterprise level
2. Resource oriented Web Service

Resource-Oriented Web Services

- Based on “resources”
- Resource -any directly accessible and distinguishable distributed component available on the network.
- RESTfulWeb Services

Architectural style in which clients and servers exchange representations of resources by using a standardized interface and protocol.

How this works



Resources

- Every distinguishable entity is a resource.
- A resource may be a Web site, an HTML page, an XML document, a Web service, an image, a video etc.

Web Services (data, functionality on server side) implemented using HTTP + REST principles

Key elements of RESTful Web service are:

- The URI (path) of the Web Service
- The HTTP method supported by the web service.
- The MIME type of the request and response data supported by it.

SOAP vs REST

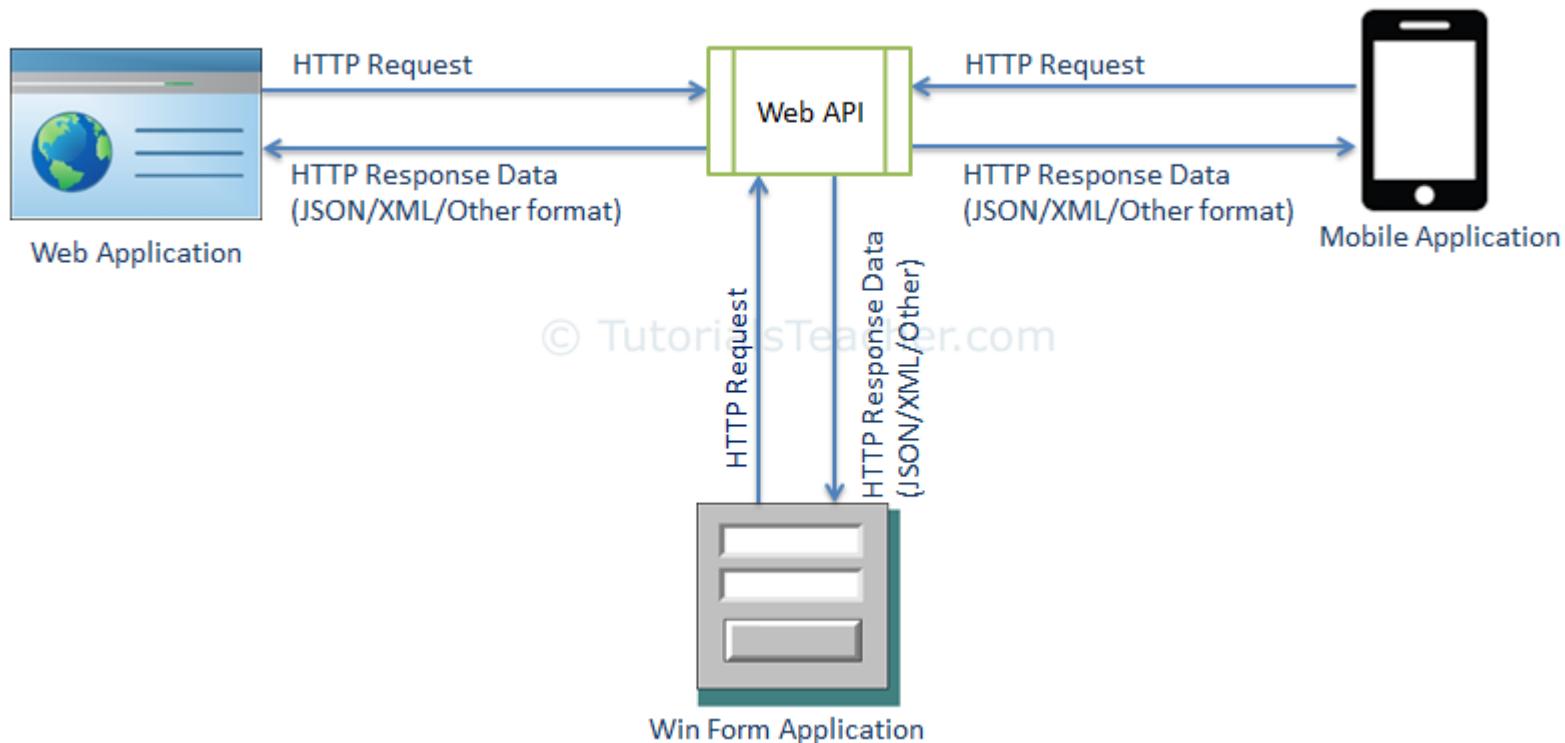
- REST is more dynamic, no need for creating and updating UDDI.
- REST is not restricted to XML format. REST web services can send plain text, JSON, and also XML.
- REST is protocol independent. It's not coupled to HTTP
- REST is as standardized as the parts you're using. Security and authentication in HTTP are standardized, so that's what you use when doing REST over HTTP.
- SOAP is a **protocol**. REST is an **architectural style**
- SOAP **can't use REST** because it is a protocol. REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
- SOAP **requires more bandwidth** and resource than REST. REST **requires less bandwidth** and resource than SOAP.
- REST has better performance and scalability. REST reads can be cached, SOAP based reads cannot be cached.

WEB - API

Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. Web API is an ideal platform for building RESTful applications

API (Application Programming Interface).

Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc.



SSL

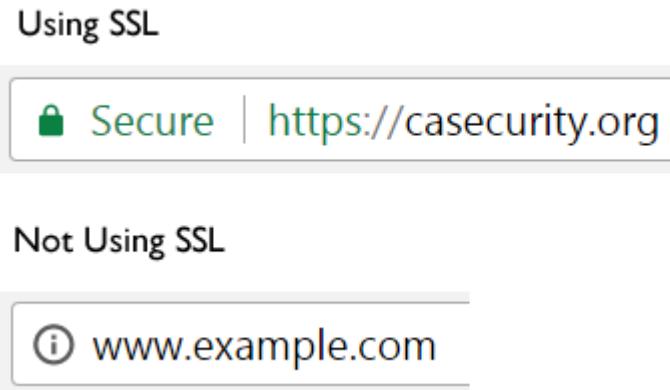
SSL (Secure Sockets Layer) is the standard **security technology** for establishing an **encrypted link** between a **web server and a browser**. This link ensures that all data passed between the web server and browsers **remain private and integral**. SSL is an **industry standard** and is used by millions of websites in the protection of their online transactions with their customers

One of the most important components of **online business** is **creating a trusted environment where potential customers feel confident in making purchases**. SSL certificates create a foundation of trust by establishing a secure connection and browsers give visual cues, such as a lock icon or a green bar, to help visitors know when their connection is secure.



How to identify the website is Secure with SSL

Technically, SSL is a transparent protocol which requires little interaction from the end user when establishing a secure session. In the case of a browser, you can tell if a site is using SSL when a padlock is displayed or the address bar shows the URL as HTTPS instead of HTTP.



With so much of our day to day transactions and communications happening online, there is very little reason for not using SSL. SSL supports the following information security principles:

Encryption: protect data transmissions (e.g. browser to server, server to server, application to server, etc.)

Authentication: ensure the server you're connected to is actually the correct server.

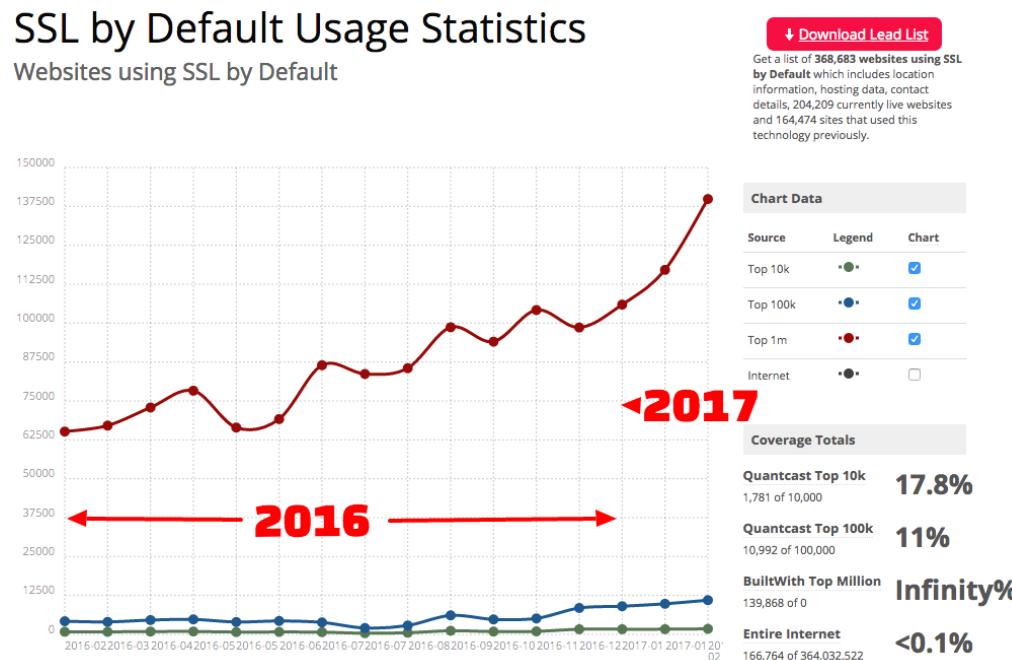
Data integrity: ensure that the data that is requested or submitted is what is actually delivered.

SSL can be used to secure:

- Online credit card transactions or other online payments.
- Intranet-based traffic, such as internal networks, file sharing, extranets and database connections.
- Webmail servers like Outlook Web Access, Exchange and Office Communications Server.

How Do I Get SSL?

To adopt SSL in your business, you should purchase an SSL Certificate.



SSL Handshake

Client Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.

Server Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings (public key), session-specific data.

Authentication and Pre-Master Secret

Client authenticates the server certificate. (e.g. Common Name / Date / Issuer) Client (depending on the cipher) creates the pre-master secret for the session, Encrypts with the server's public key and sends the encrypted pre-master secret to the server.

Decryption and Master Secret

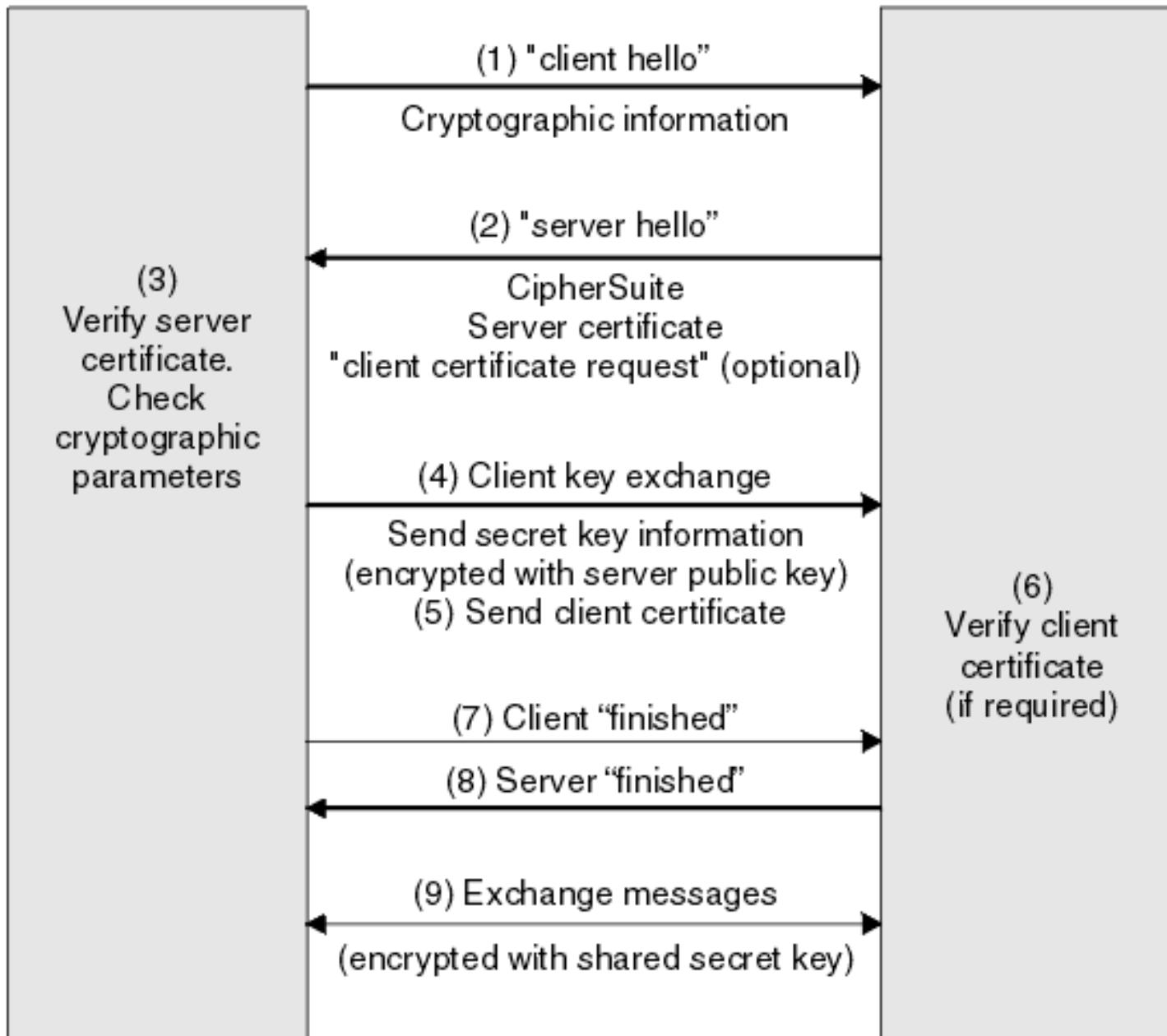
Server uses its private key to decrypt the pre-master secret. Both Server and Client perform steps to generate the master secret with the agreed cipher.

Encryption with Session Key

Both client and server exchange messages to inform that future messages will be encrypted

SSL Client

SSL Server



OpenID

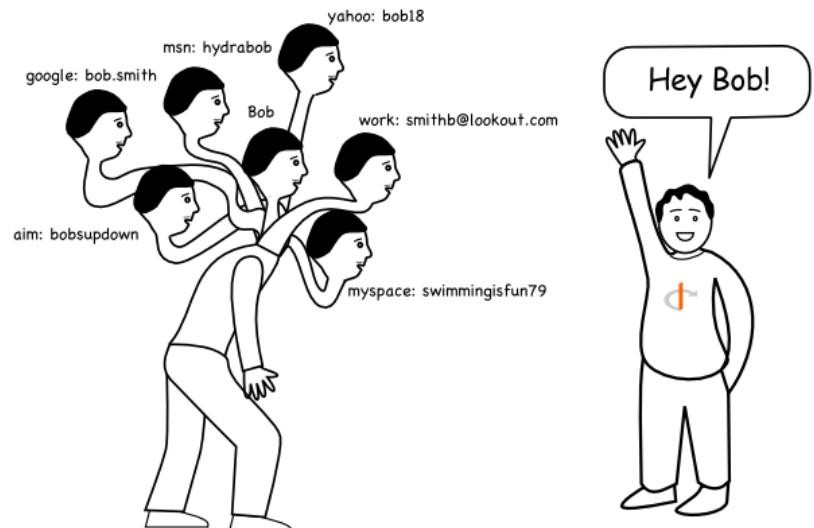
OpenID allows you to use an existing account to sign in to multiple websites, without needing to create new passwords.

You may choose to associate information with your OpenID that can be shared with the websites you visit, such as a name or email address. With OpenID, you control how much of that information is shared with the websites you visit.

With OpenID, your password is only given to your identity provider, and that provider then confirms your identity to the websites you visit. Other than your provider, no website ever sees your password, so you don't need to worry about an unscrupulous or insecure website compromising your identity.

**Promoted by the non-profit OpenID Foundation, it allows users to be authenticated by co-operating sites –
(WIKIPEDIA)**

OpenID also simplifies signing in. With OpenID **you only have to remember one username and one password**



OpenID vs OAuth

OpenID is about authentication (ie. proving who you are), OAuth is about authorization (ie. to grant access to functionality/data/etc.. without having to deal with the original authentication).

First the scenario for OpenID:

- User wants to access his account on example.com
- example.com (the “Relying Party” in OpenID lingo) asks the user for his OpenID
- User enters his OpenID
- example.com redirects the user to his OpenID provider
- User authenticates himself to the OpenID provider
- OpenID provider redirects the user back to example.com
- example.com allows the user to access his account

And now the scenario for OAuth:

- User is on example.com and wants to import his contacts from mycontacts.com
- example.com (the “Consumer” in OAuth lingo) redirects the user to mycontacts.com (the “Service Provider”)
- User authenticates himself to mycontacts.com (which can happen by using OpenID)
- mycontacts.com asks the user whether he wants to authorize example.com to access his contacts
- User makes his choice
- mycontacts.com redirects the user back to example.com
- example.com retrieves the contacts from mycontacts.com
- example.com informs the user that the import was successful

OAuth

Used for delegated **authorization** only -- meaning you are authorizing a third-party service access to use personal data, without giving out a password. Also OAuth "sessions" generally live longer than user sessions. Meaning that OAuth is designed to allow authorization

i.e. Flickr uses OAuth to allow third-party services to post and edit a persons picture on their behalf, without them having to give out their flicker username and password.

OpenID

Used to **authenticate** single sign-on identity. All OpenID is supposed to do is allow an OpenID provider to prove that you say you are. However many sites use identity authentication to provide authorization (however the two can be separated out)

i.e. One shows their passport at the airport to authenticate (or prove) the person's who's name is on the ticket they are using is them.

OpenID Foundation

The OpenID Foundation (OIDF) promotes and enhances the OpenID community and technologies. The OIDF is a non-profit international standards development organization of individual developers, government agencies and companies who wish to promote and protect OpenID. The OpenID Foundation was formed in June 2007

Questions ?

Isuru.wijeratna@ifsworld.com



Using IoT in Enterprise Applications

Enterprise Application
Development



USING IOT IN ENTERPRISE APPLICATIONS

ENTERPRISE APPLICATION DEVELOPMENT

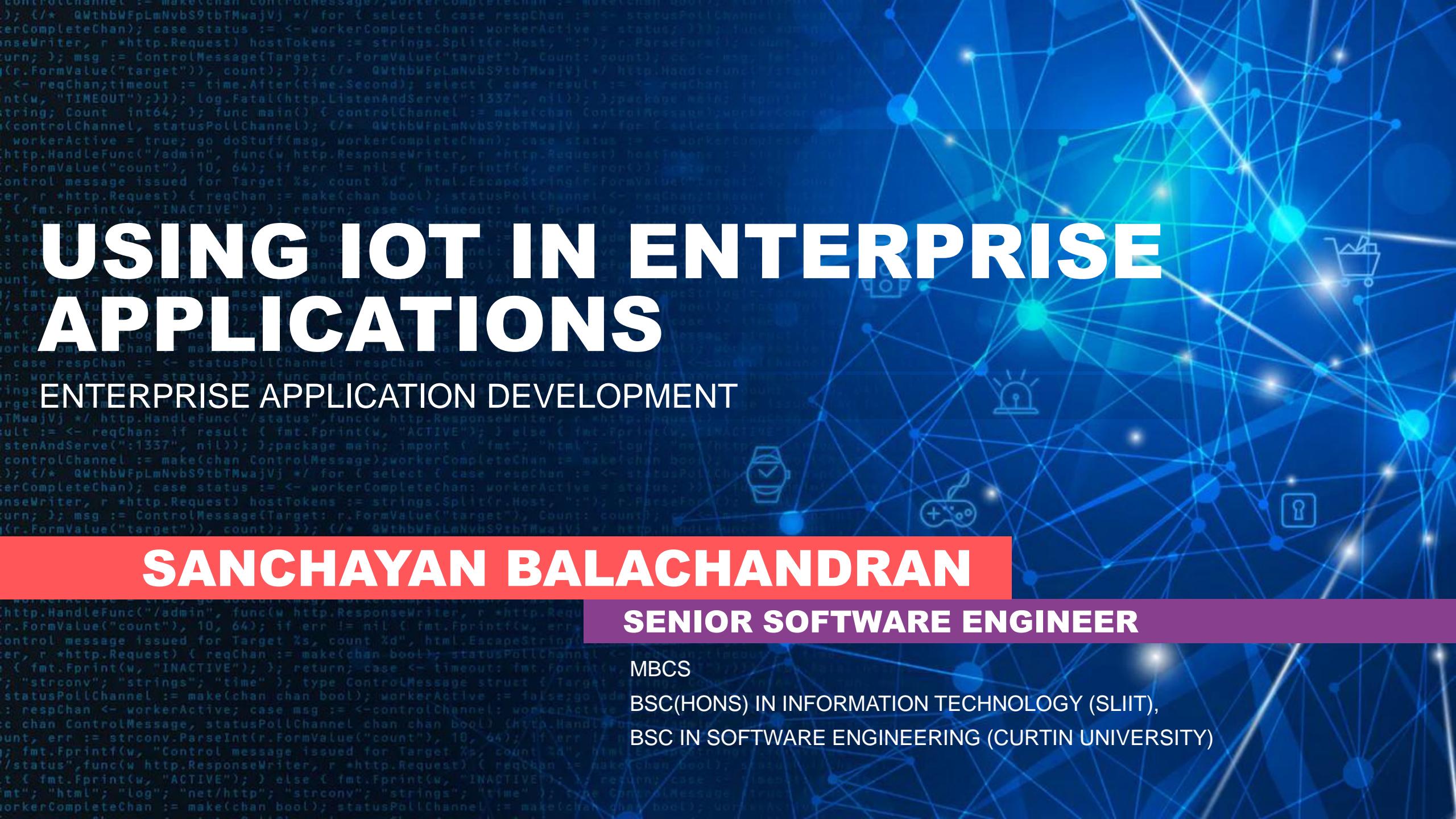
SANCHAYAN BALACHANDRAN

SENIOR SOFTWARE ENGINEER

MBCS

BSC(HONS) IN INFORMATION TECHNOLOGY (SLIIT),

BSC IN SOFTWARE ENGINEERING (CURTIN UNIVERSITY)



Agenda

01 Overview - IoT

02 IoT in ERP

03 IoT Architecture

04 Demo

What is IOT

Internet of Things



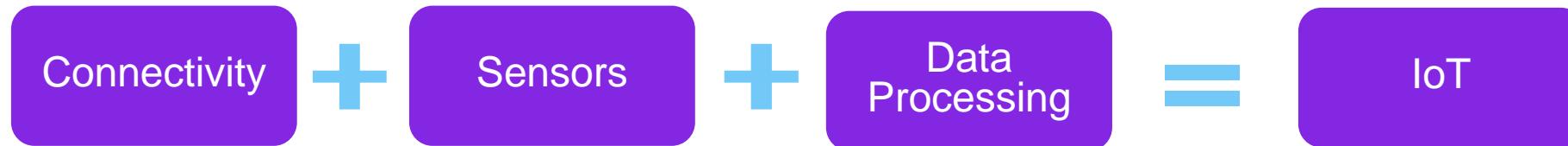
Definition

Internet of Things

- The **Internet of Things (IoT)** is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data, creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions

General Idea

Internet of Things



Add connectivity to basic functionality

Add smart sensors to increase functionality



New Services based on the technological possibilities

Application Types

Internet of Things

Consumer Applications

- Smart home
- Elder care



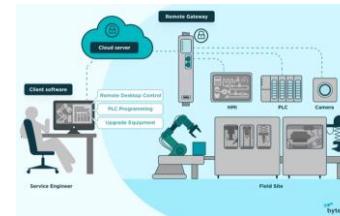
Commercial Applications

- Healthcare
- Transportation



Industrial Applications

- Manufacturing
- Supply Chain



Infrastructure Applications

- Metro services
- Energy Management



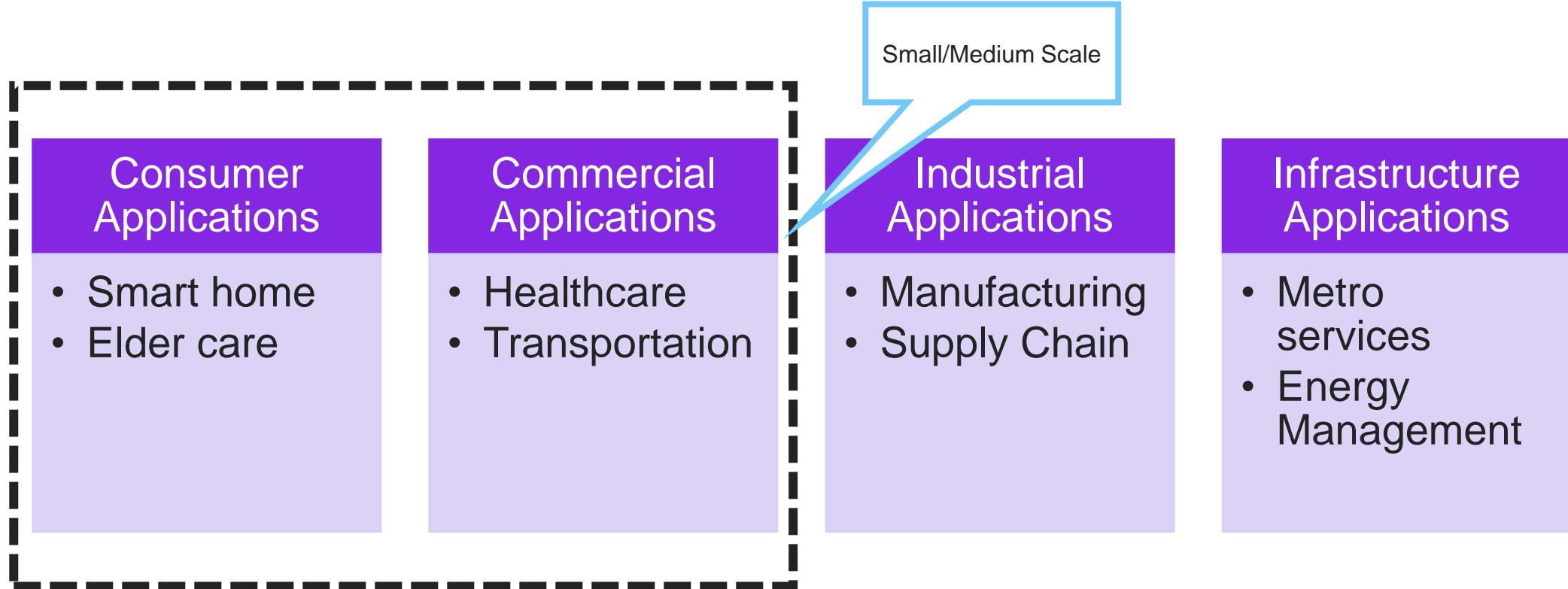
Application Types - Example

Internet of Things

- Smart Home: smart lighting system, thermostats, smart gardening
- Elder care: wearable GPS tracker
- Health care: Smart continuous glucose monitoring
- Transportation: Traffic management, Connected cars (Tesla maintenance)
- Manufacturing: Remote production control, logistics tracking, predictive repairing
- Supply Chain: Warehouse management
- Metro services: Repair maintenance (fewer maintenance delays)
- Energy Management: Monitor devices and alert

Application Types

Internet of Things



Application Types

Internet of Things

Consumer Applications

- Smart home
- Elder care

Commercial Applications

- Healthcare
- Transportation

Industrial Applications

- Manufacturing
- Supply Chain

Enterprise Scale

Infrastructure Applications

- Metro services
- Energy Management



ERP Operational Flow in General

What is an ERP?

- Enterprise resource planning is the integrated management of core business processes, often in real-time and mediated by software and technology



Business Processes

IOT can be used in...

- Service & Maintenance
- Manufacturing
- Supply Chain
- Field Service Management
- Oil & Gas Industry
- HR





Example Service & Maintenance

Process

Fault in Equipment

- Someone in work floor identifies fault



Process

Fault in Equipment

- Nature of fault being communicated to an officer/admin



Process

Fault in Equipment

- Maintenance/Repair order being created

The screenshot shows the IFS Applications software interface. The top navigation bar includes 'IFS Navigator', 'Financials', 'Accounts Receivable', and 'Customer Analytics'. The left sidebar contains icons for 'Report Library', 'Customer Analysis', 'General Ledger', 'Banking', 'Marketing', 'Project Accounting', 'CRM', and 'Purchasing'. The main window displays a 'Customer Details' screen for 'Customer 2000'. It shows a balance of 290.00 EUR and an advance balance of 49.00. Below this are sections for 'Tax Withholding', 'Bills', and 'Reminder Templates'. A large grid table lists transaction details, including columns for 'Serial ID', 'No.', 'Period', 'Amount', 'Debit', 'Credit', 'Date', 'Description', and 'Status'. The bottom of the screen shows a toolbar with various icons and a status bar indicating 'Ready'.

Process

Fault in Equipment

- Order report generated and handed over to service engineer



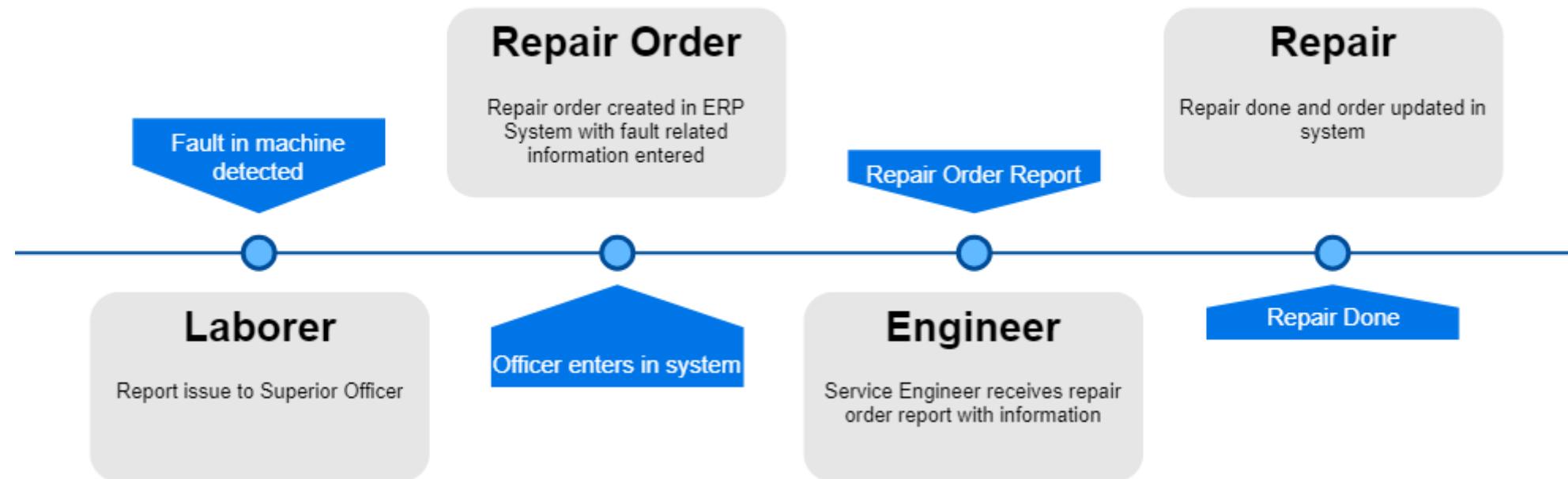
Process

Fault in Equipment

- Reports back in system once repair is done



A Typical Business Process



“Do you see anything wrong here?”

Lot of Manual Work

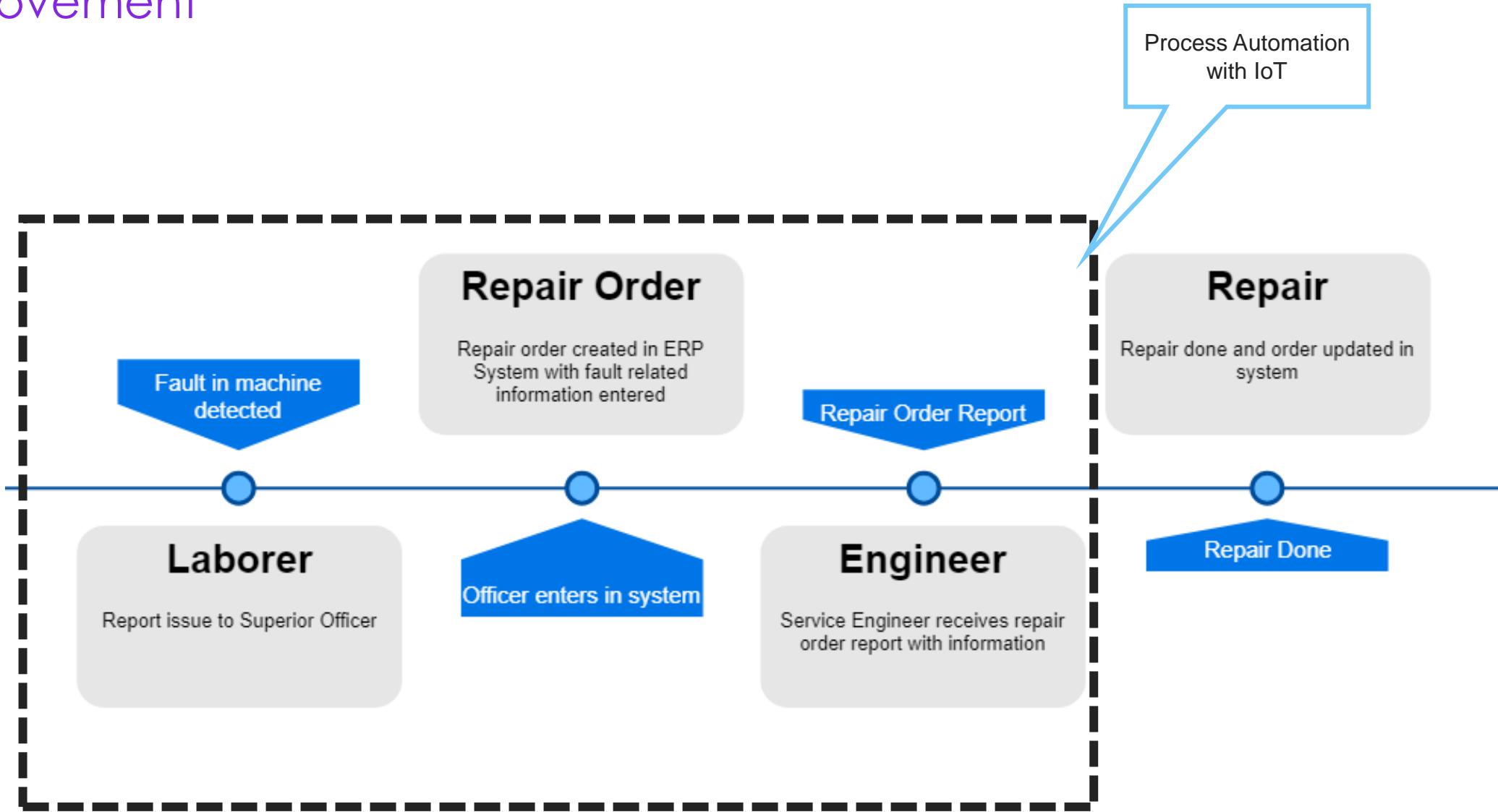


Room for Improvement

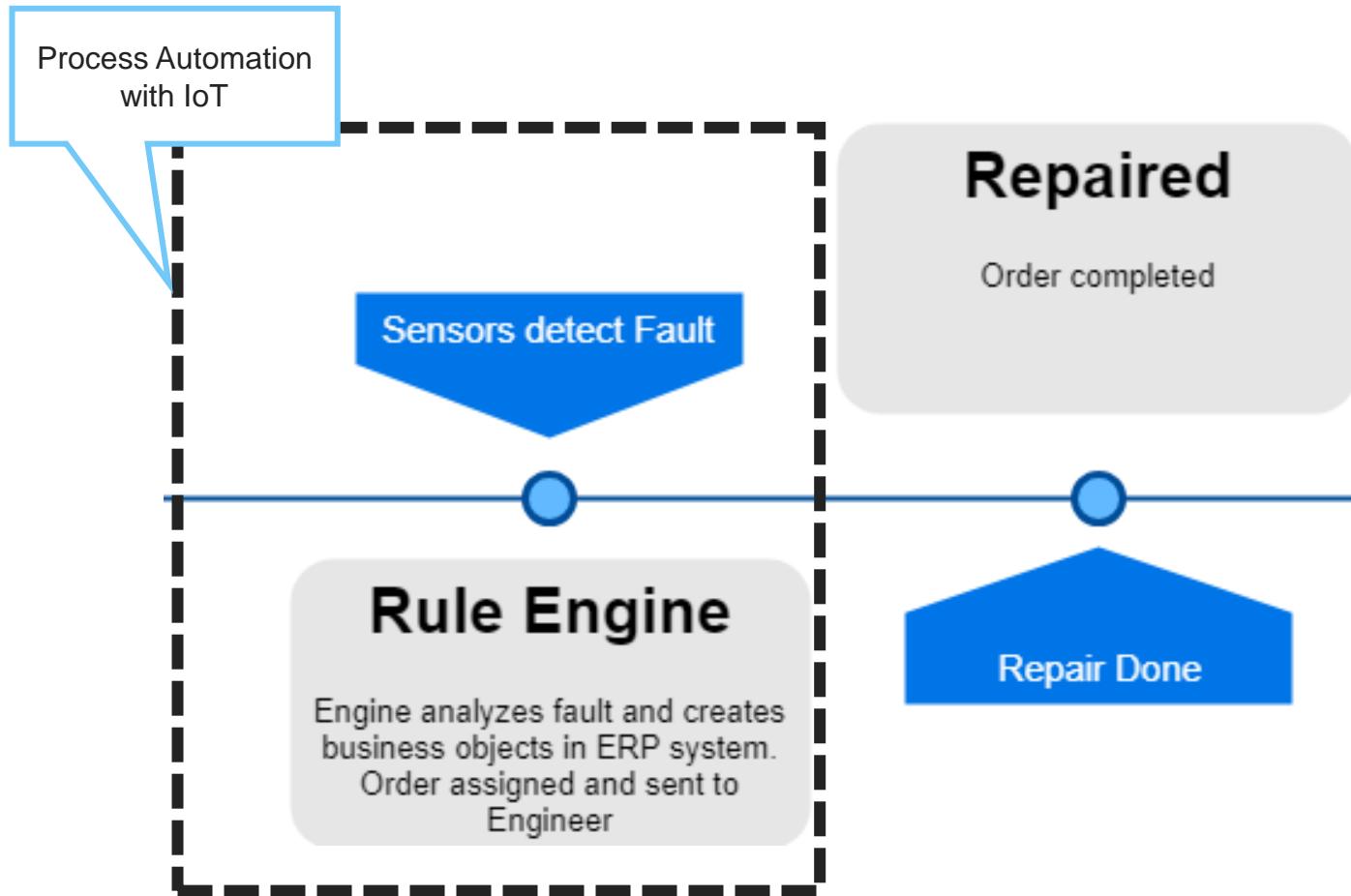
- Efficiency
- Productivity
- Cost
- Scalability
- Automation

“Is IoT the answer?”

Process Improvement



Process Improvements

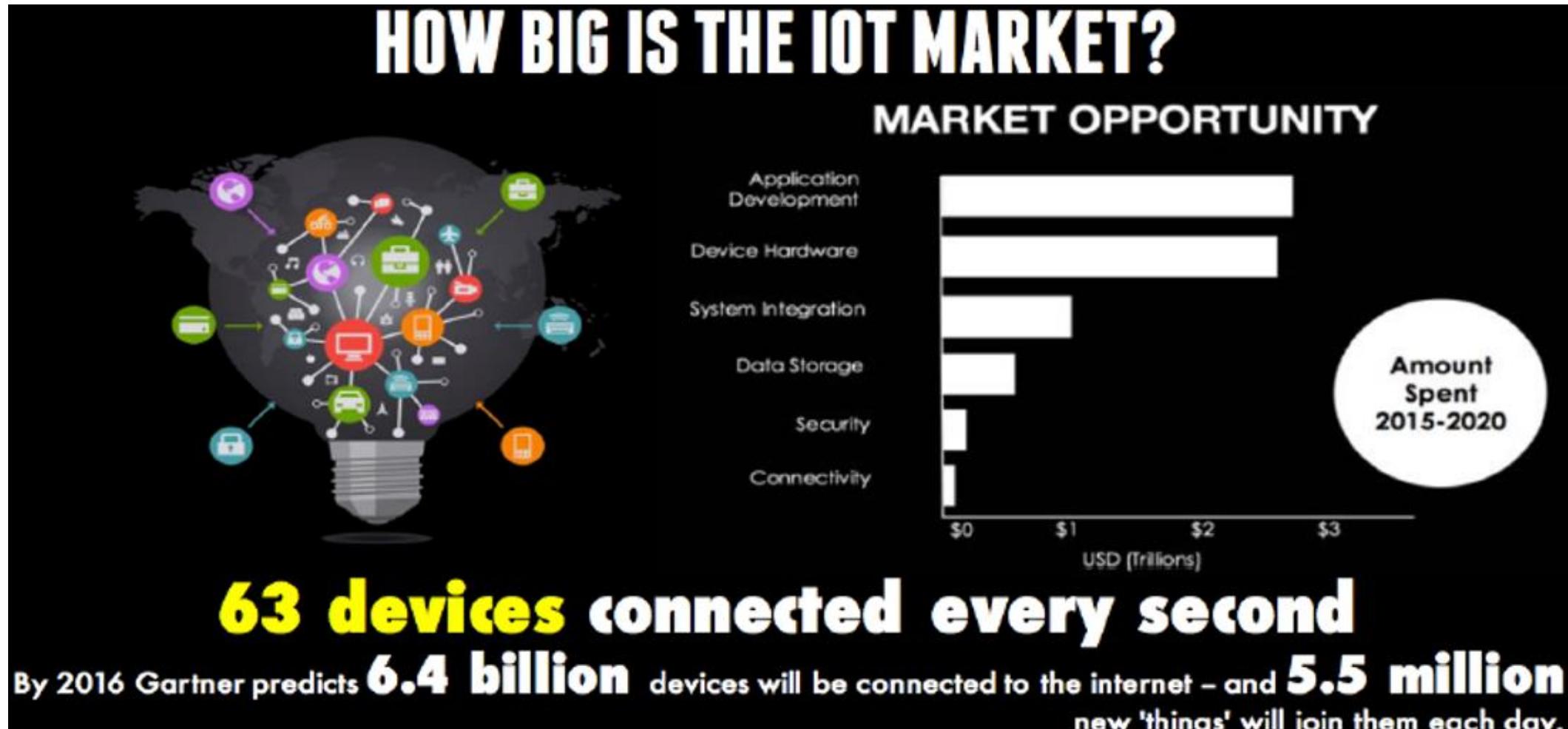




IOT & ERP Market Reach & Potential

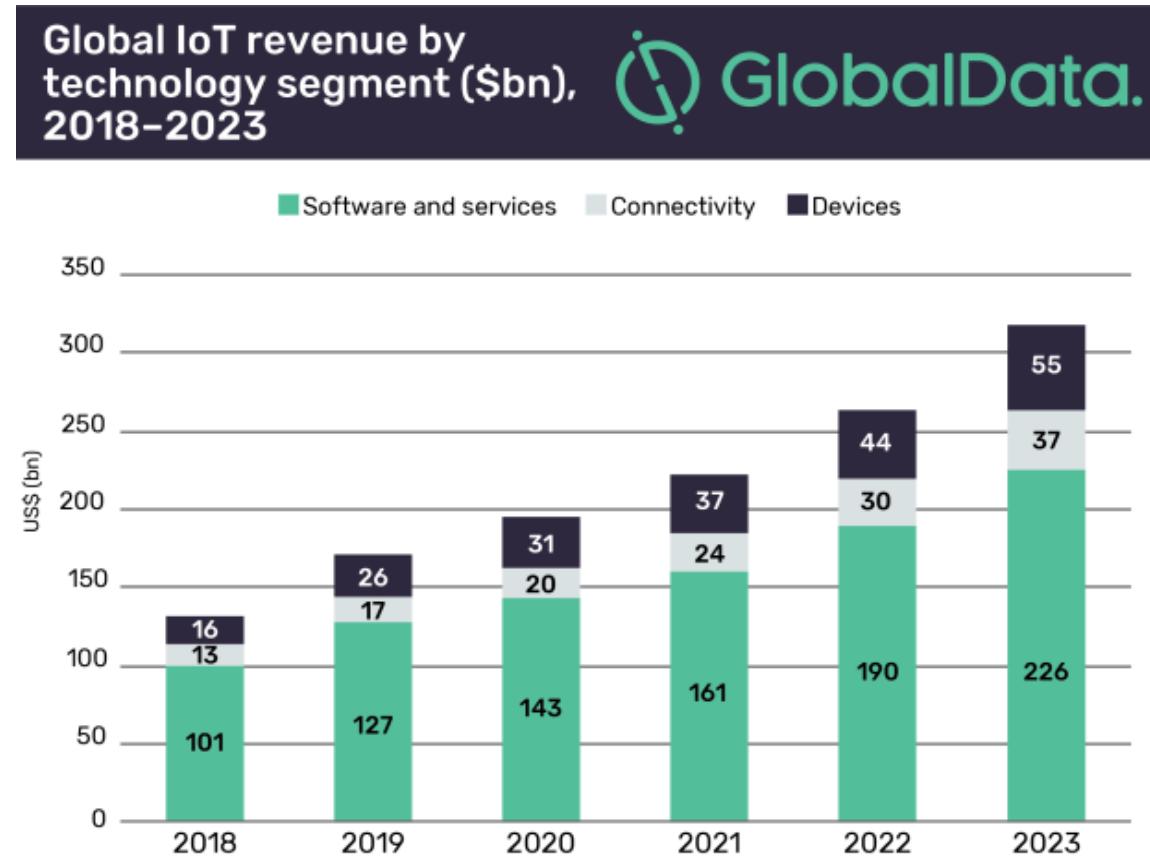
Market Reach

IOT



Market Reach

IOT



Source: GlobalData, Technology Intelligence Centre

Areas of Interest



MANUFACTURING

35% of manufacturers already use smart sensors. 10% plan to implement them within a year, and 8% plan to implement them within 3 years, according to PwC



OIL, GAS, AND MINING

In five years, it is predicted that 5.4 million IoT devices will be used on oil extraction sites. BI Intelligence said that these devices will primarily be internet-connected sensors used to provide environmental metrics about extraction sites.



TRANSPORTATION

Connected cars are a top IoT device. We estimate there will be over 220M connected cars on the road by 2020.



INSURANCE

A survey has found that 74% of insurance executives believe the IoT will disrupt insurance within the next five years. 74% also plan to invest in developing and implementing IoT strategies by 2016, according to an SMA Research survey.

[Source: <http://www.cbronline.com/news/internet-of-things/m2m/smart-connected-always-on-10-huge-iot-impacts-on-10-huge-sectors-4709736v>]

Areas of Interest



DEFENSE

We estimate spending on drones will reach \$8.7B in 2020. In addition, 126K military robots will be shipped in 2020, according to Frost & Sullivan.



CONNECTED HOME

smart meters are already a reality, and from the 313 million installed in homes worldwide in 2013, Navigate Research predicts that number to jump to 1.1 billion by 2022



AGRICULTURE

We estimate 75M IoT devices will be shipped for agricultural uses in 2020, at a 20% CAGR. These devices are primary sensors placed in soil to track acidity levels, temperature, and variables that help farmers increase crop yields.



FOOD SERVICES

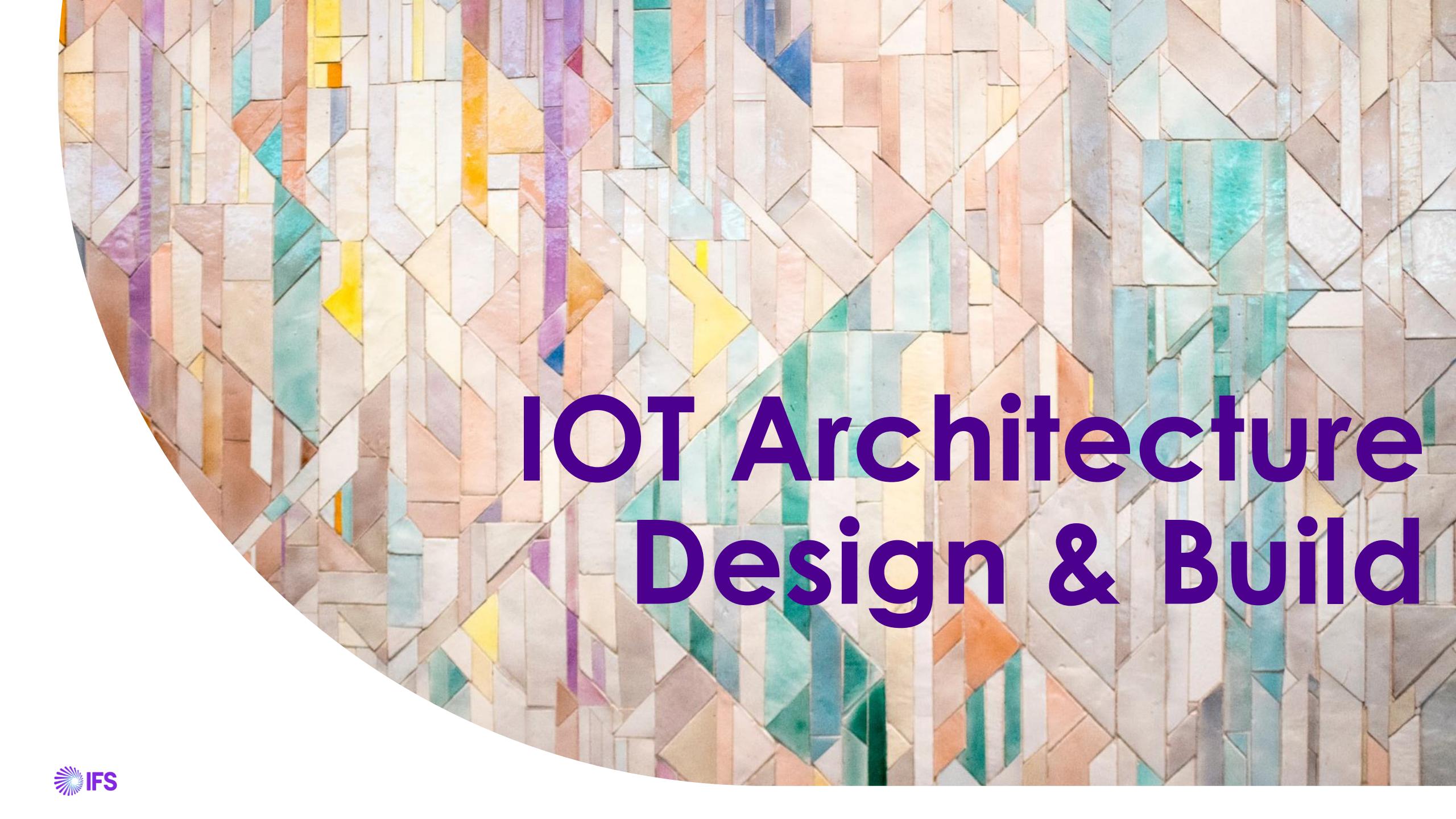
With the world population on the rise, IoT will help to keep food production levels at good pace. BI Intelligence estimates 75 million IoT device installations for agricultural uses by 2020, at a 20% CAGR.

Source: <http://www.cbronline.com/news/internet-of-things/m2m/smart-connected-always-on-10-huge-iot-impacts-on-10-huge-sectors-4709736v/>

What is different for ERP?

Challenges

- Security risks have much bigger consequences compared to consumer devices
 - Strict laws would mean misuse or any breach of data would mean serious legal issues
- Downtime can disrupt the whole system and daily operations which could result in loss of millions of dollars
 - Time critical operations such as maintenance and repair in large scale industries can take a bigger hit even for lesser amount of downtime
- Cost of running things
 - Maintaining hardware and equipment along with cloud resources needed to handle IoT operations would mean substantial additional cost
- Latency can be more of a problem in some business cases
 - Can be an issue when sensors produce large volume of data
- Connectivity issues due to large volume of data
 - Can be an issue when assets are in remote locations where network coverage is poor

The background of the slide features a complex, abstract geometric pattern composed of numerous overlapping triangles. These triangles are oriented in various directions and are filled with a variety of pastel-colored gradients, including shades of pink, purple, yellow, orange, teal, and light blue. The overall effect is a vibrant, modern, and dynamic visual texture.

IOT Architecture Design & Build

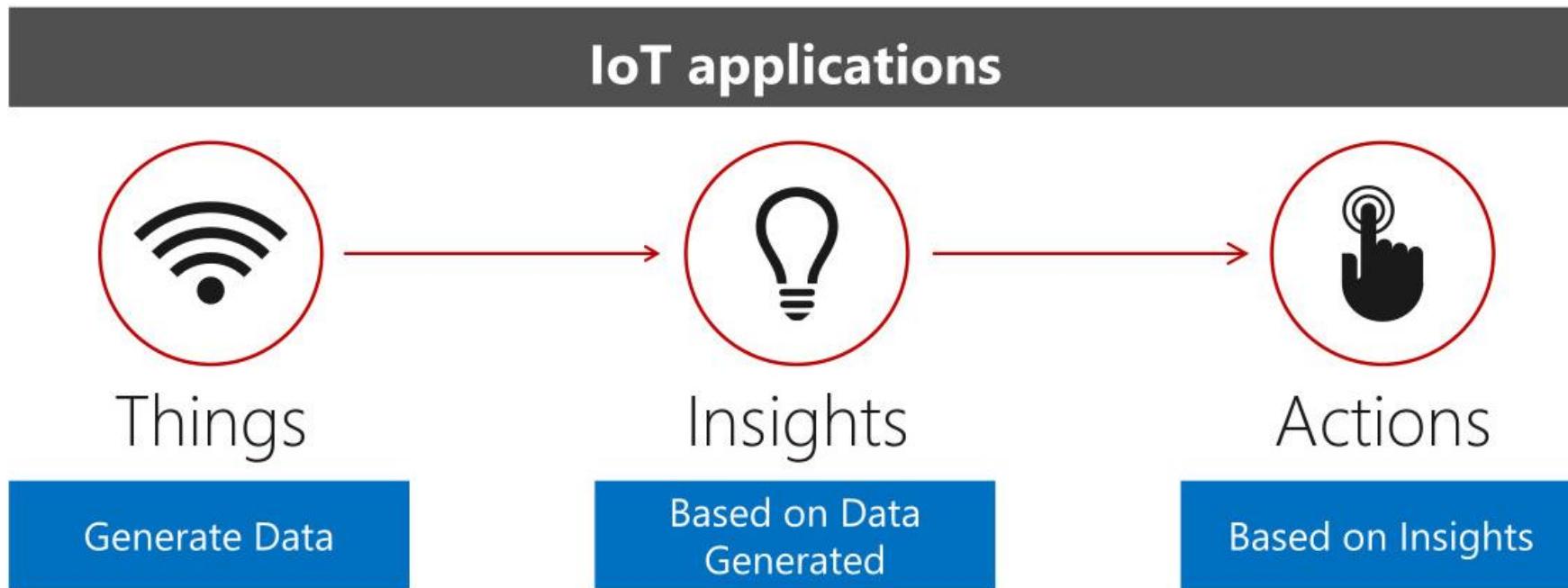
What to Keep in Mind

IOT Architecture

- Security
- Simplicity
- Performance
- Scale
- Flexibility
- Cost

Main Goals

IOT Architecture



- Every organization has unique skills and experience
- Every IoT application has unique needs and considerations

Core Design

Main Components

- Devices (and/or on-premise edge gateways)
 - Component that has the ability to securely register with the cloud, and connectivity options for sending and receiving data with the cloud



Core Design

Main Components

- Hubs
 - a cloud gateway service, or hub, to securely accept that data and provide device management capabilities.



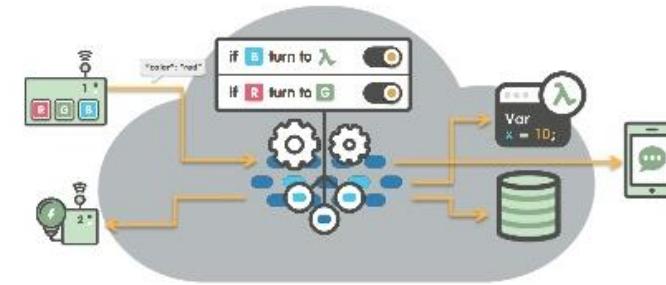
Core Design

Main Components

- Engine
 - **Stream processors** that consume that data, integrate with **business processes**, and place the data into **storage**. Makes it possible to build IoT applications that gather, process, analyze and act on data generated by connected devices at global scale without having to manage any infrastructure



Azure Stream Analytics



AWS IoT Rules Engine

Core Design

Main Components

- Storage
 - Storage can be divided into warm path (data that is required to be available for reporting and visualization immediately from devices), and cold path (data that is stored longer term and used for batch processing)



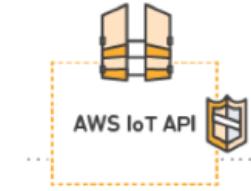
Azure Service Bus Queue



Core Design

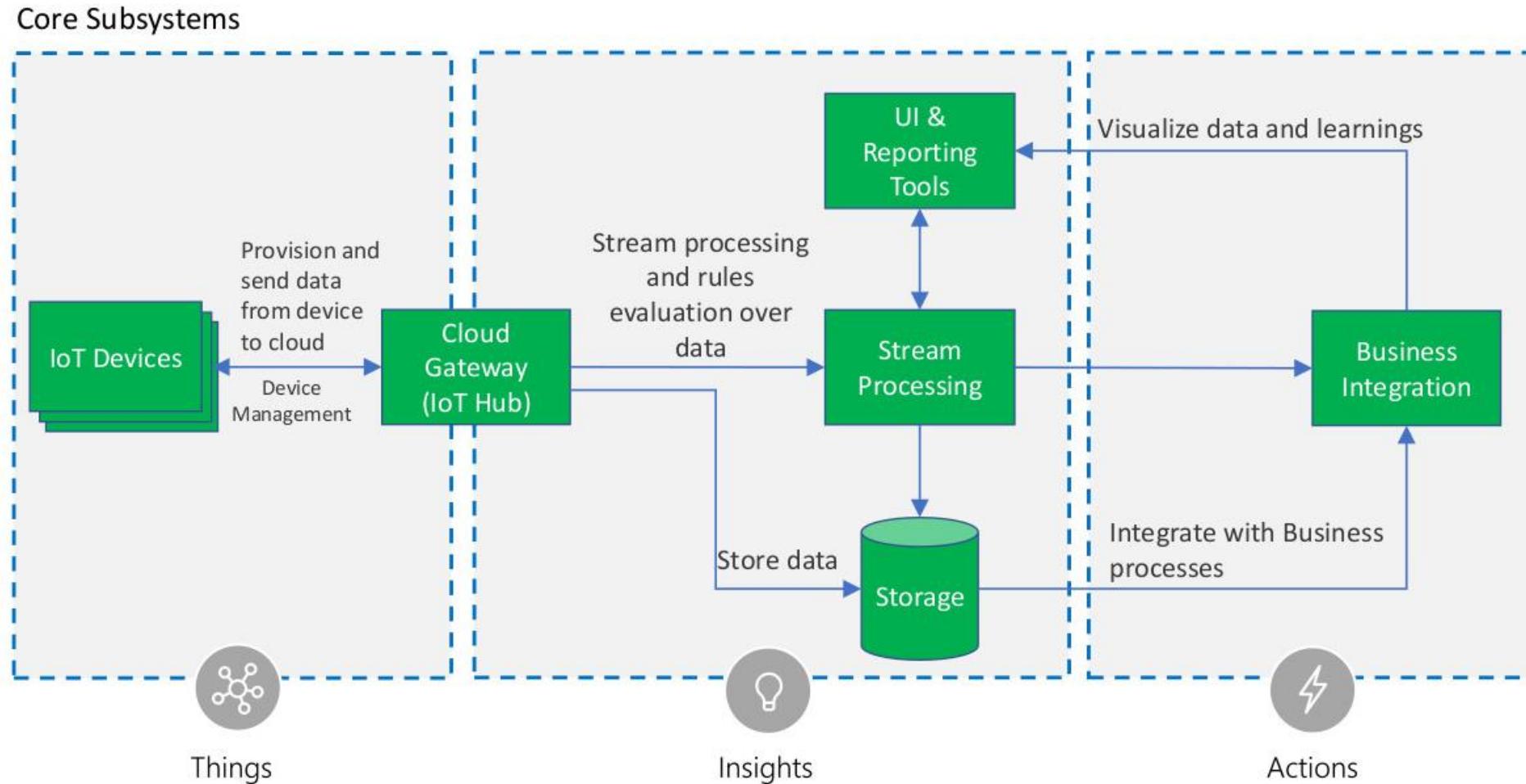
Main Components

- Business Integration
 - Integration facilitates executing actions based on insights garnered from device telemetry data during stream processing. Integration could include storage of informational messages, alarms, sending email or SMS, integration with CRM, and more



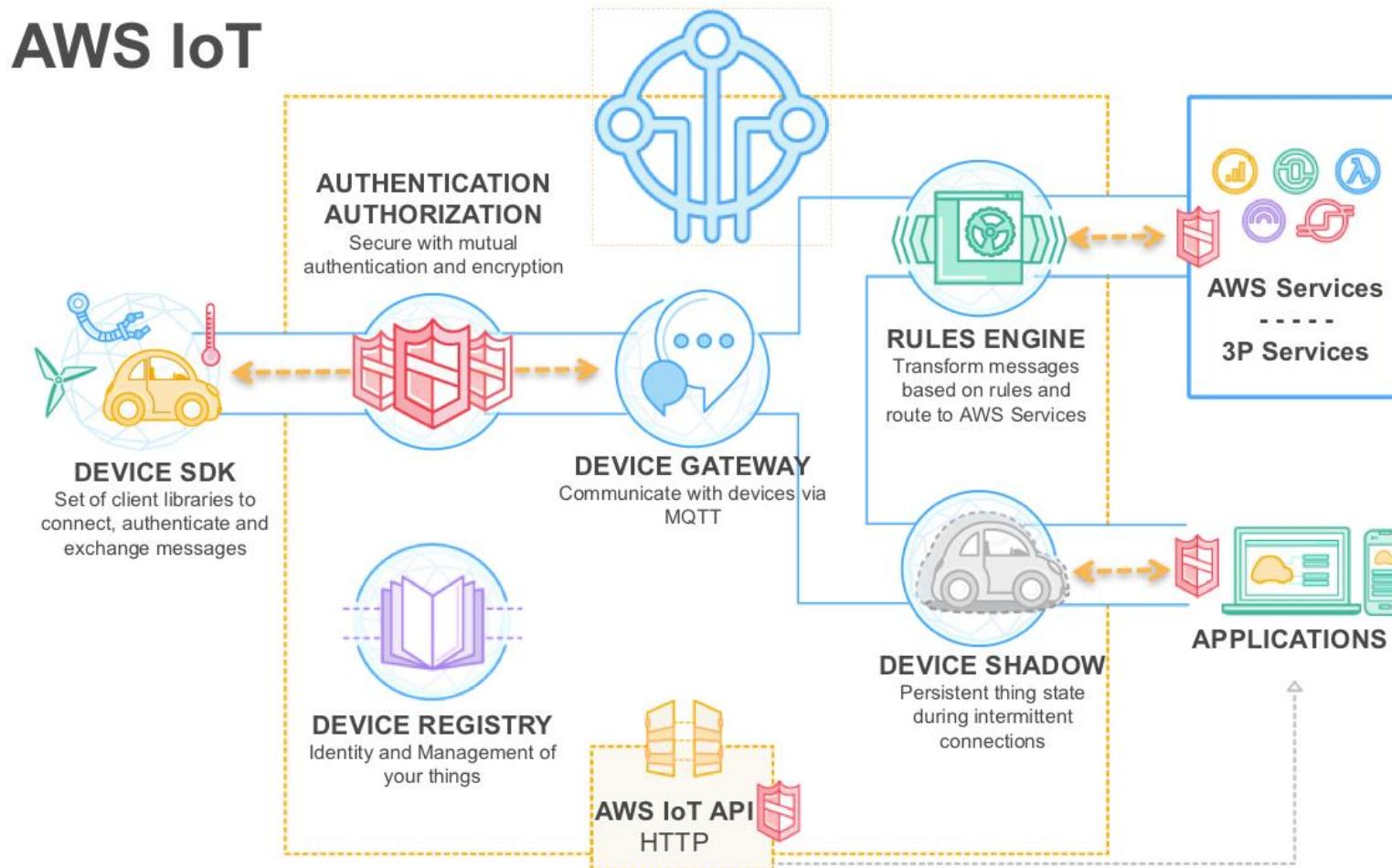
Reference Architecture

Azure



Reference Architecture

AWS



Things to Keep in Mind

IOT Reference Architecture

- Reference architecture is just *recommended* ways to do this
- Choose the most suitable components for the business need
- Keep up to date on latest IoT offerings
- Design can always be improved



Upcoming Trends



Tackling Challenges in ERP World

Cost Problem

- One of the biggest challenges in designing an IoT application in enterprise scale is the cost of running things. Running high-cost resources in this scale to manage devices would mean more cost for both customer and the application vendor.
- Cost of cloud offerings can be a big burden to bear for ERP vendors and customers if the scale of application is too big

Possible Solution

Cost

- The architectural designs in industry should find a balance between speed and cost depending on customer needs. For time critical operations high sophisticated CPU intensive resources will be used. For less time critical low-cost storage type resources will be used.

But How?



Hot Path vs Cold Path

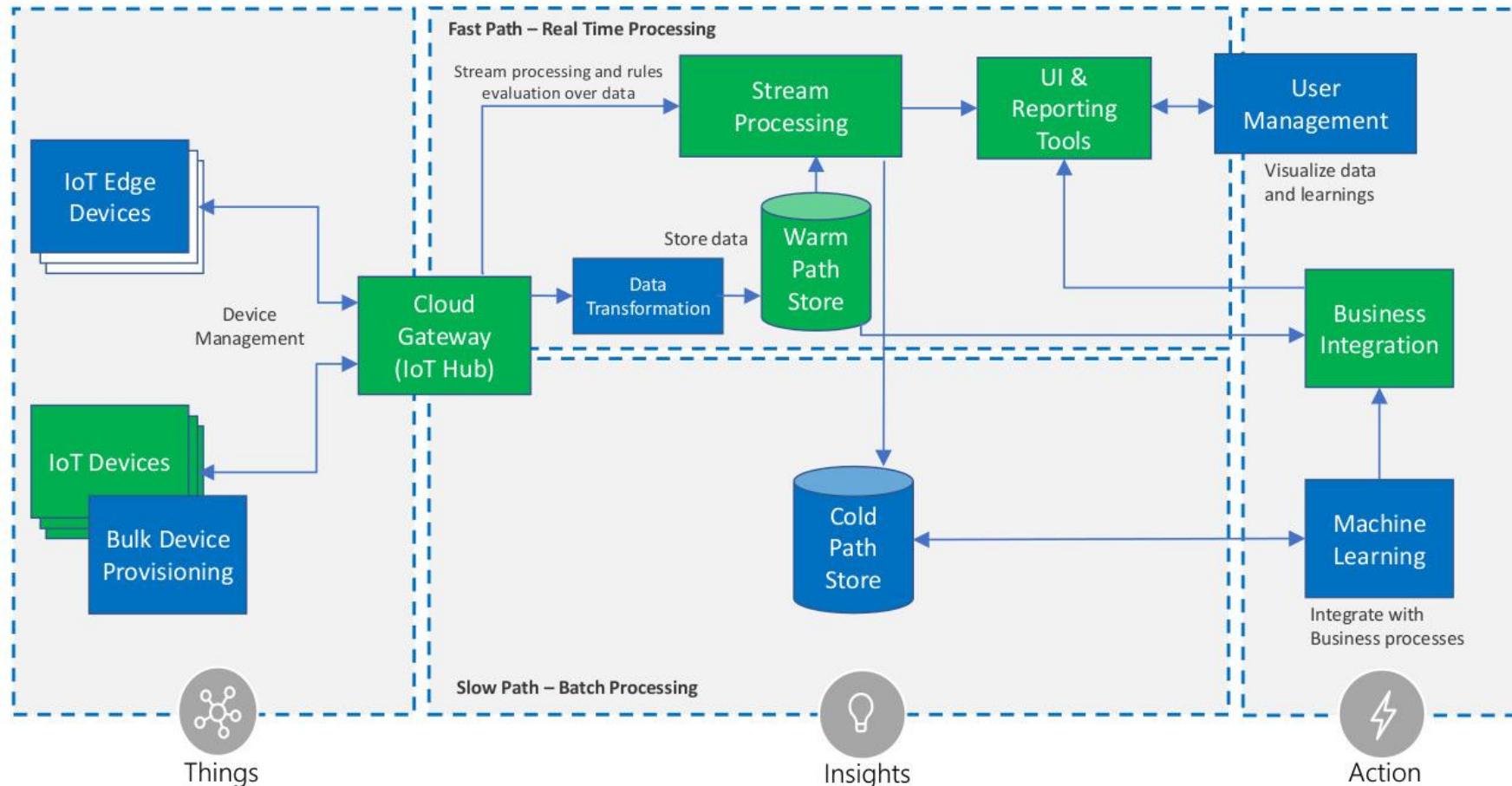
What is this

Hot Path Vs Cold Path

- Hot path:
 - (near) Real time data flow
 - Fast results
 - Time critical
- Cold path:
 - Non real time data
 - Batch data flow
 - Slow results
 - Minutes maybe hours
 - System integration

Hot Path Vs Cold Path

All Subsystems – Lambda Architecture



Other Problems

- Latency
 - The delay before a transfer of data begins following an instruction for its transfer.
- Privacy
 - How to handle sensitive data

Possible Solution

- **Edge Computing**
 - Many refer to this trending paradigm by the term ***Edge Computing*** as a way to emphasize that part of the work happens right at the edge of the network where IoT connects the physical world to the Cloud. But Edge Computing is much more than having computation and data processing on IoT devices. A fundamental part of it is the strong and seamless integration between IoT and Cloud; between the physical world and the world of computation.

Edge Computing Application

Definition

- An Edge Computing application is an application that uses the processing power of IoT devices to filter, pre-process, aggregate or score IoT data. It uses the power and flexibility of Cloud services to run complex analytics on those data and, in a feedback loop, support decisions and actions about and on the physical world.

Why Edge Computing

Motivating Factors

Preserve privacy

Data captured by IoT devices can contain sensitive or private information, e.g., GPS data, streams from cameras, or microphones. While an application might want to use this information to run complex analytics in the Cloud, it is important that, whenever data leaves the premises where it is generated, the privacy of sensitive content is preserved. With Edge Computing, an application can make sure that sensitive data is pre-processed on-site, and only data that is privacy compliant is sent to the Cloud for further analysis, after having passed through a first layer of anonymizing aggregation.

Why Edge Computing

Motivating Factors

Reduce latency

The power and flexibility of Cloud computing has enabled many scenarios that were impossible before. Think about how the accuracy of image or voice recognition algorithms has improved in recent years. However, this accuracy has a price: the time needed to get an image or a piece of audio recognized is significantly affected by the non-negligible yet unavoidable network delays due to data being shipped to the Cloud and results computed and sent back to the edge. When low-latency results are needed, Edge Computing applications can implement machine-learning algorithms that run directly on IoT devices, and only interact with the Cloud off the critical path, for example, to continuously train machine learning models using captured data.

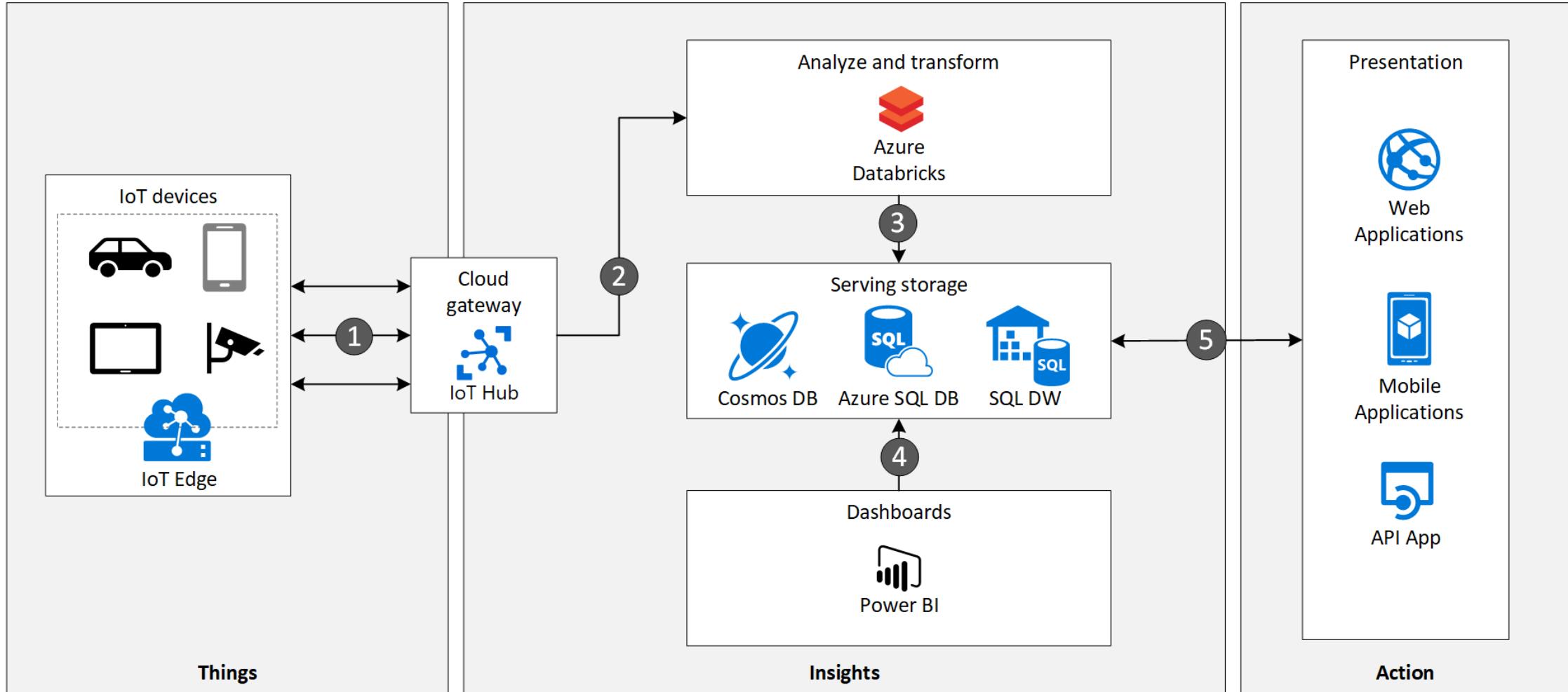
Why Edge Computing

Motivating Factors

Be robust to connectivity issues

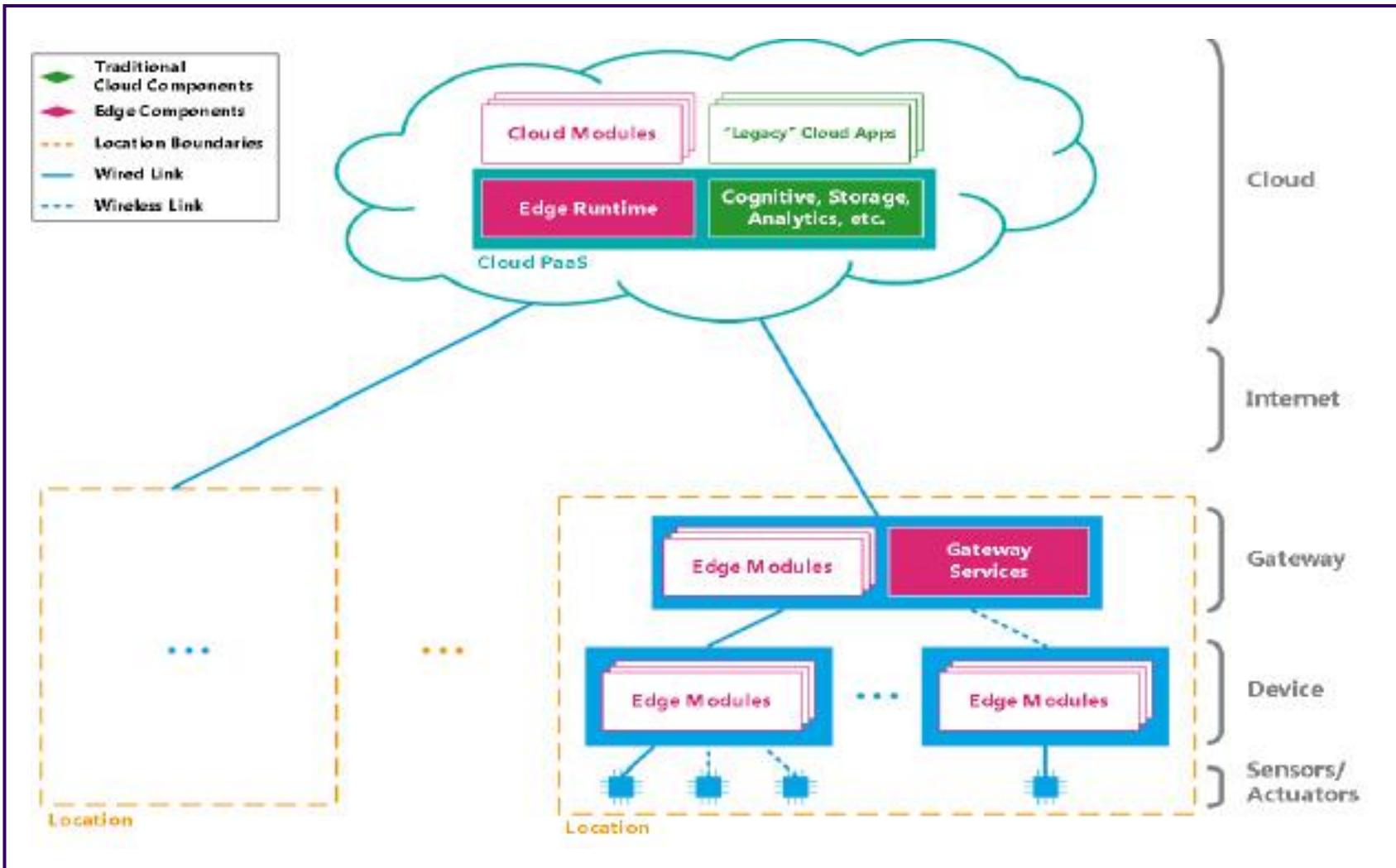
Designing applications to run part of the computation directly on the Edge not only reduces latency, but potentially ensures that applications are not disrupted in case of limited or intermittent network connectivity. This can be very useful when applications are deployed on remote locations where network coverage is poor or even to reduce costs coming from expensive connectivity technologies like cellular technologies.

IOT Edge



IOT Edge

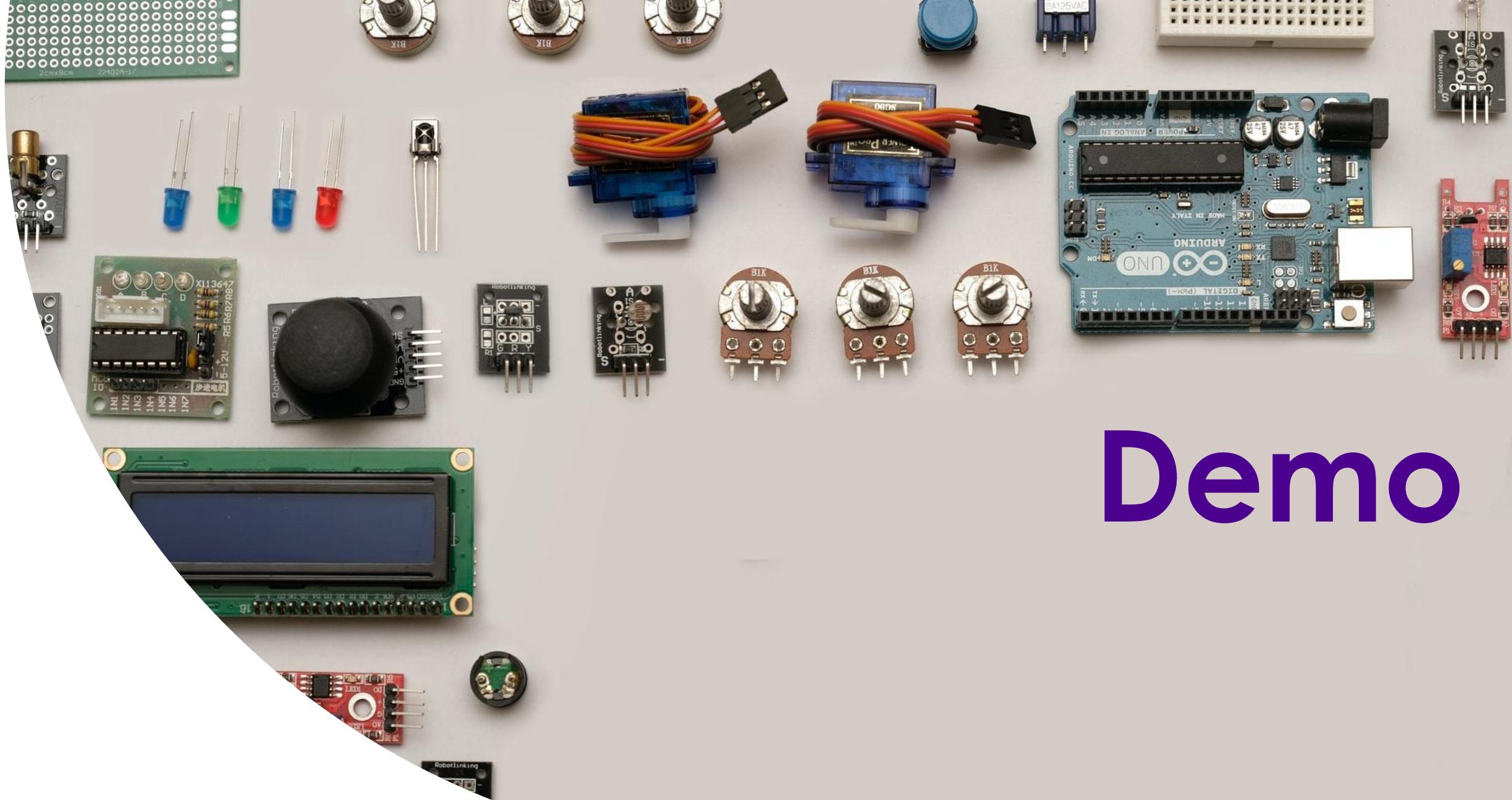
Reference Architecture



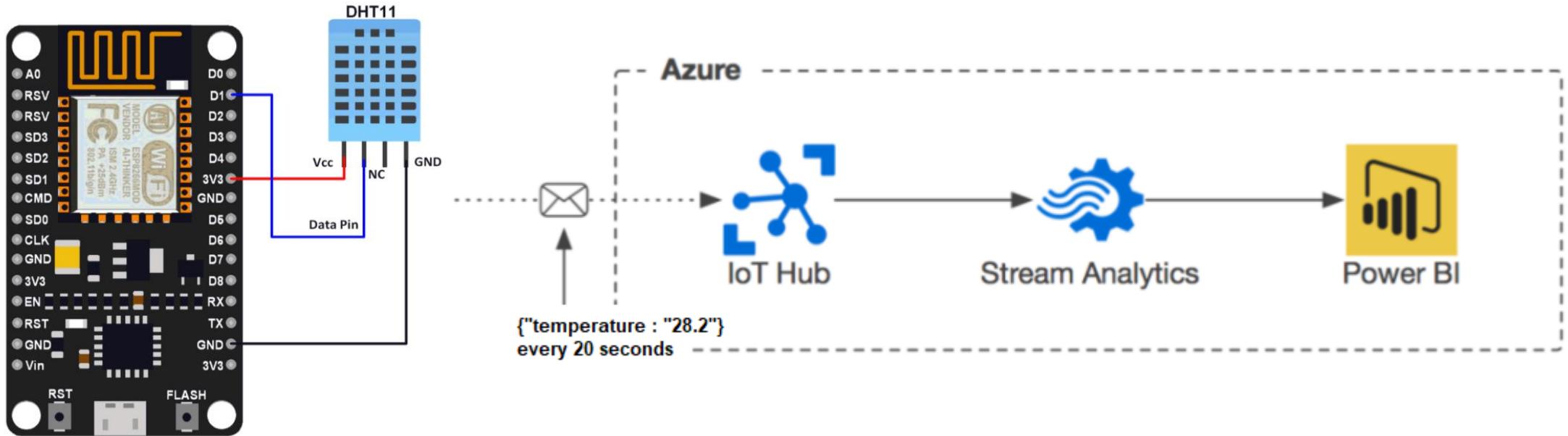


Questions?

Demo



Demo

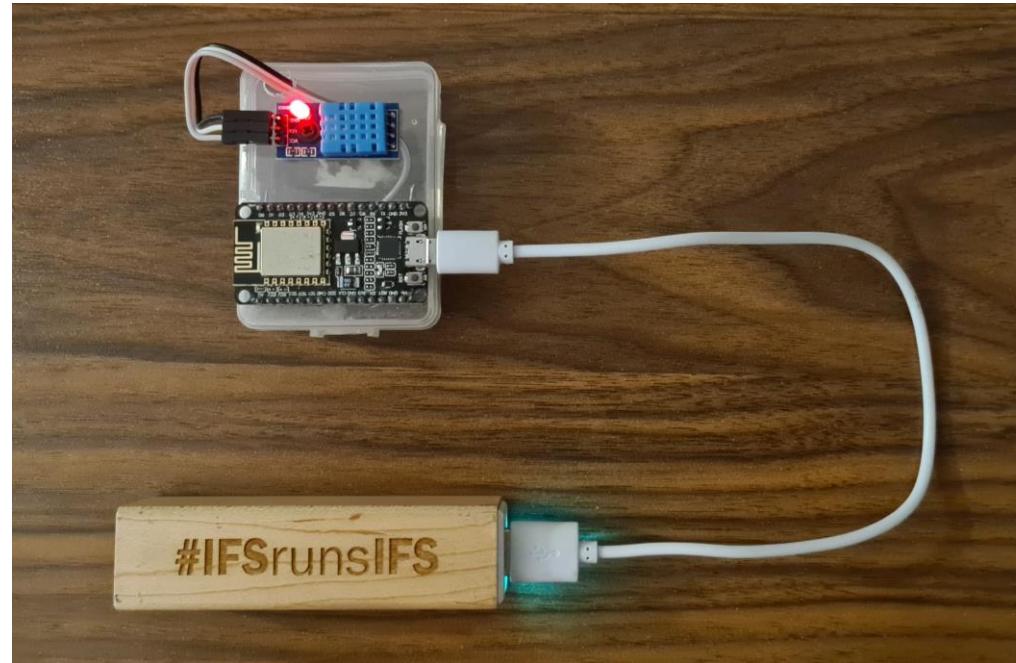


Demo

Setup

References

- Sending DHT11 sensor data to IoT Hub using NodeMCU
<https://www.electronicwings.com/azure/sending-dht11-sensor-data-to-iot-hub-using-nodemcu>



Thank you!

#MOMENTOFSERVICE





© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.

Enterprise Application Development **Microservices**

Thilina Weliwita



Welcome

- Timings
- Safety
- Ground rules
- Objectives



Agenda

- | | | | |
|----|---|----|---|
| 01 | Monolithic Application Architecture | 06 | Communication between Microservices |
| 02 | Microservices Architecture | 07 | Advantages & Disadvantages of Microservices |
| 03 | Monolith vs Microservices Architecture | | |
| 04 | Characteristics of Microservices Architecture | | |
| 05 | Microservices vs SOA | | |

Safety and Ground Rules

- Fire Alarm – process – exits – muster
- Phones off or on silent
- Be social, switch off Social Media
- Respect
- Access



Handouts

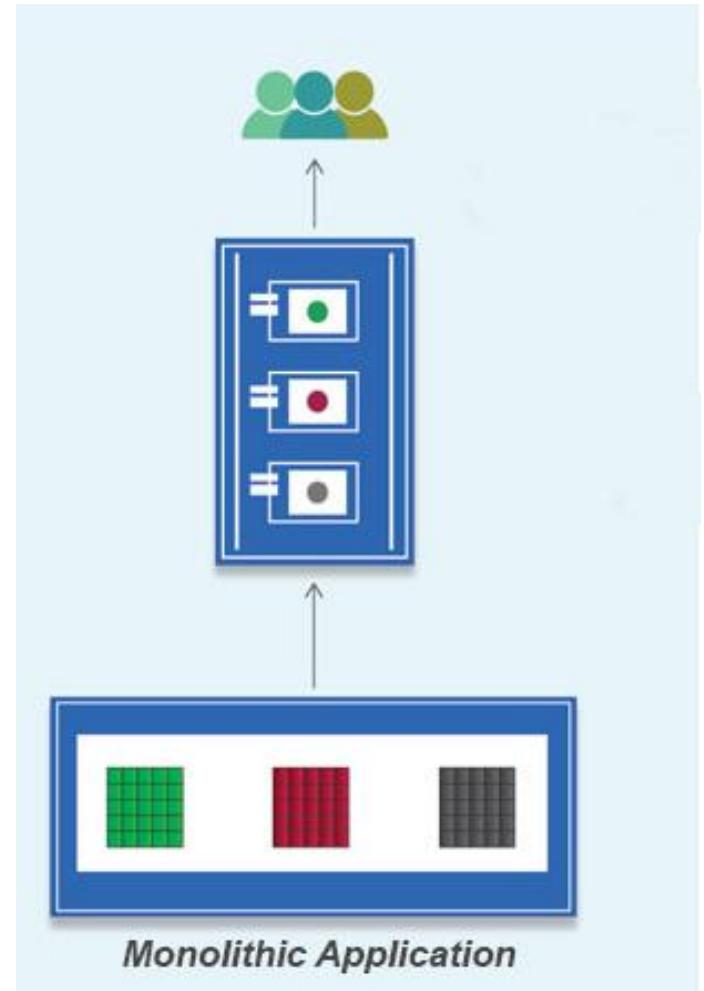


01. Monolithic Application Architecture

Monolithic Application Architecture

Monolithic Application

- All business capabilities / modules will run in a single process.
This may be a WAR / EAR file.
- Classes of each of those business capabilities / modules may communicate with each other and may share same libraries.
- One module may be a library or dependency for another module.
- Connects to one single database containing tables related to all business capabilities



Monolithic Application

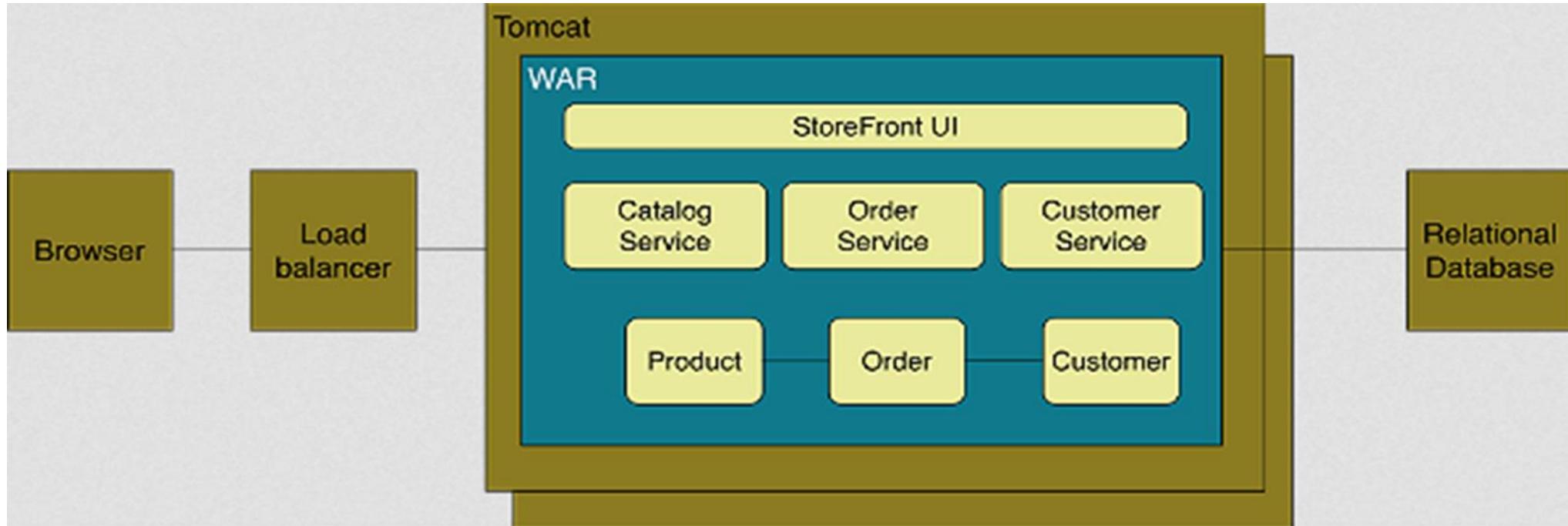
Architecture

- A Monolithic application is built as a single unit.
- It includes all its functionality into a single process.
- May contain 3 tiers such a client tier, server (middle/business) tier and database tier.

Monolithic Application

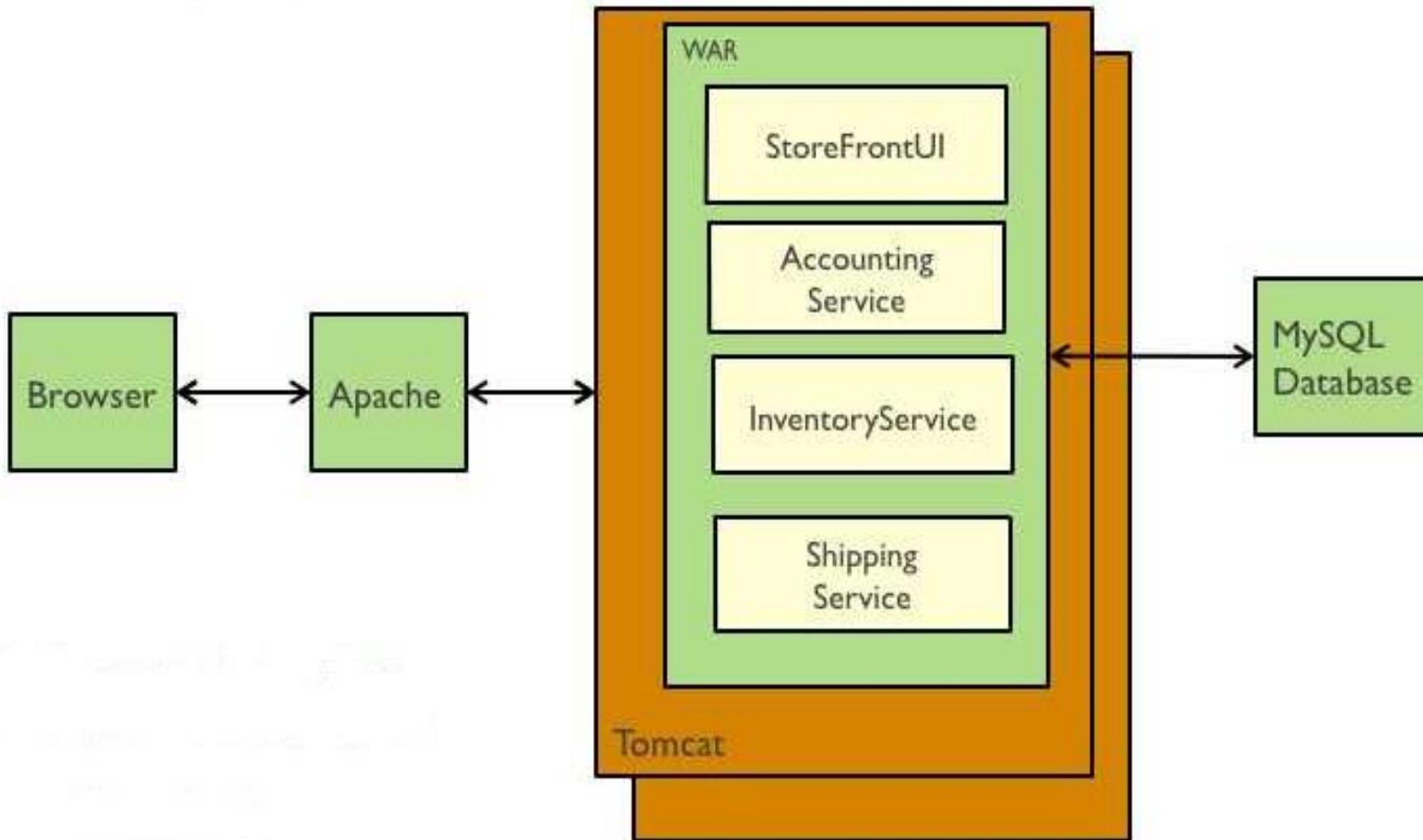
Architecture

- The Server Tier may contain all the business logic which spans across various business domains across the application.



Monolithic Application

Architecture



Monolithic Application

Architecture

- These business logic may be written in classes, and they may be grouped into namespaces or packages according to the business capabilities or technical boundaries.

```
package com.example.shoppingcartservice
```

```
package com.example.usermanagement
```

Monolithic Application

Architecture

- Function in a particular business domain may call another function of a different business domain.

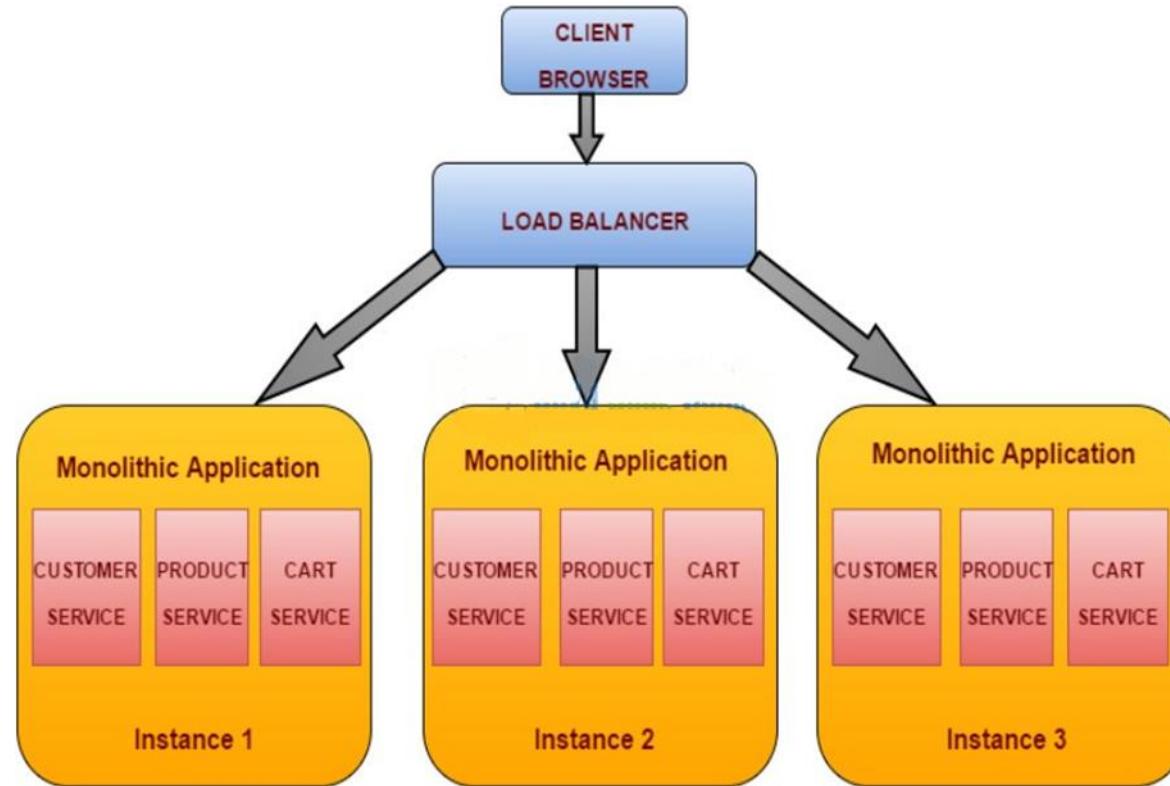
```
9  
10 public class ProductCatalog {  
11  
12     Public Product viewProduct(int id) {  
13         ...  
14         ...  
15         RecommendationEngine recEngine = new RecommendationEngine();  
16         recEngine.getRecommendedProducts(category);  
17         ...  
18         ...  
19     }  
20 }
```

- This introduces coupling and interdependencies among the business modules of the enterprise application.

Monolithic Application

Architecture

- Monolithic application can be horizontally scaled by running many instances behind a load balancer, but each instance contain all functionalities even though there are components having less demand.



Monolithic Application

Architecture

- Monolithic application can be hosted in different geographical locations.



- However, the entire application modules has to be deployed even though some of the modules do not have high demand.

Monolithic Application

Architecture

- Monolithic architecture approach suits most of the software applications but have become inadequate for some large enterprise applications.
- Some of the major worldwide business companies such as Netflix and Amazon has migrated from Monolithic architecture to Microservice architecture.

Advantages of Monolithic Architecture

- Simplicity
- Easier code Base Access
- Easier inter- module refactoring
- Easier Deployments

Monolithic Application Architecture

Simplicity

- Monolith is relatively a simple and familiar architecture.
- Will suit for most of the simple and enterprise applications as well and we have been using it to solve the problems with the help of SOA.
- Can be implemented with small amount of resources (people & technology).
- If properly designed there will be no (or less amount of) redundant classes.
- Solution for the entire application will be one single unit (JAR / WAR / EAR).

Monolithic Application Architecture

Easier Code Base Access

- All business logic will reside in one single place.
- No need to maintain different or multiple version control systems to maintain the codebase.

Monolithic Application Architecture

Easier Inter-Module Refactoring

- You can do changes to the code base which could affect multiple modules.

(You can move classes from one module to another)

Monolithic Application Architecture

Easier Deployments

- Final delivery of a Monolithic application server will most probably be a single deployment unit such as a JAR, WAR or EAR file.
- So you just have to worry about deploying one single file.

Note: Some may argue that it is easier to deploy microservices since it concerns about one single business capability. So there is nothing right or wrong about these arguments since it depends on the task and the expectation of the person.

02. Microservices Architecture

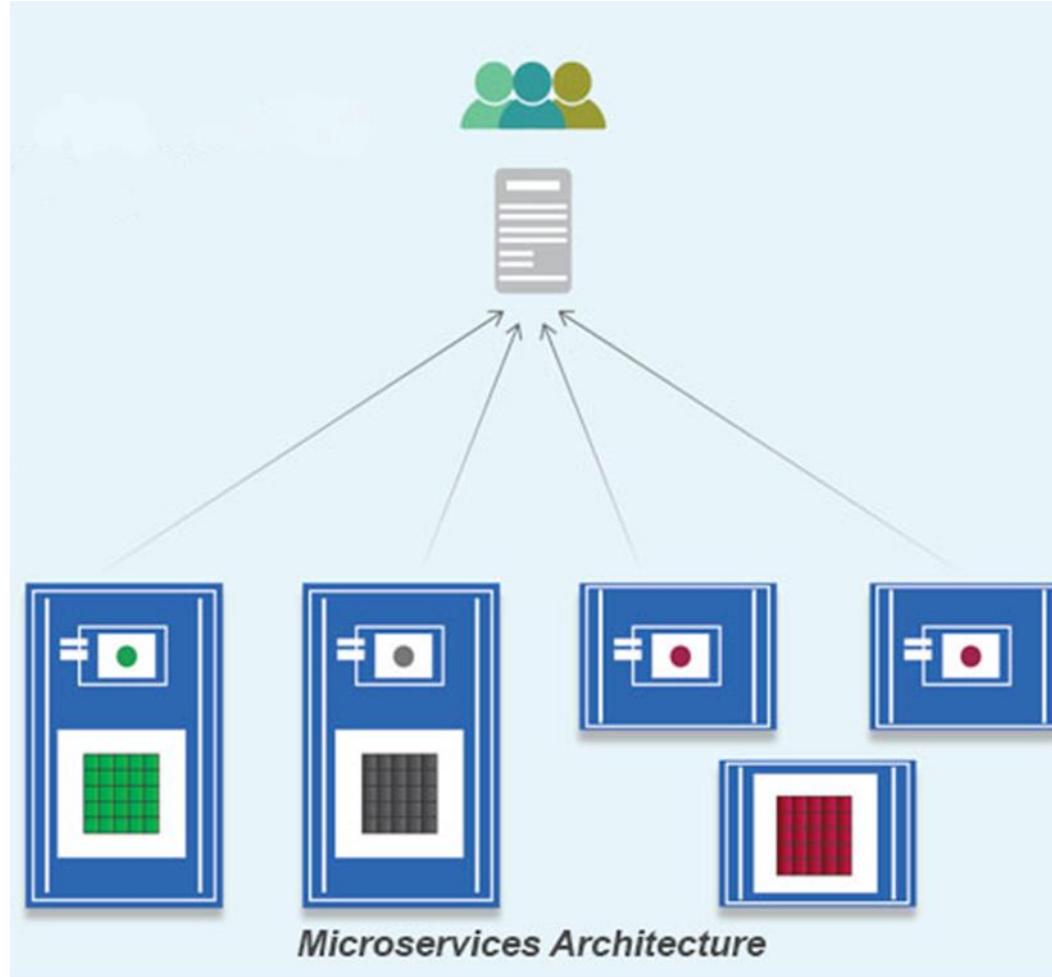
Microservices

Overview

- A self-contained process that provides a unique business capability.
- An approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.
- Built around business capabilities and independently deployable.
- Each Microservice may be written in different programming languages and use different data storage technologies.
- Microservices communicate with each other through a well-defined interface such as REST or Message Queues.
- Every Microservice is responsible for its own data model and data.
- Microservice splits individual capabilities of a Monolithic architecture and implement within its own process.

Microservices

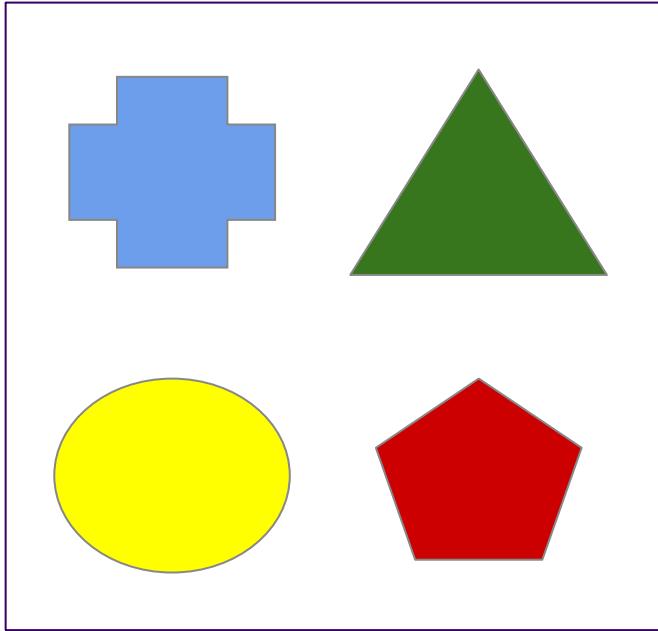
Architecture



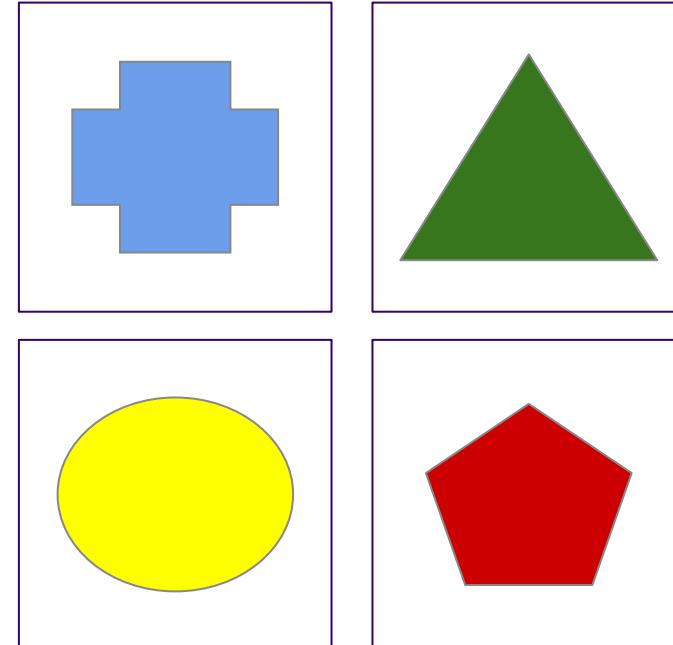
03. Monolith vs Microservices Architecture

Monolith vs Microservices

A monolith application puts all its functionality into a single process

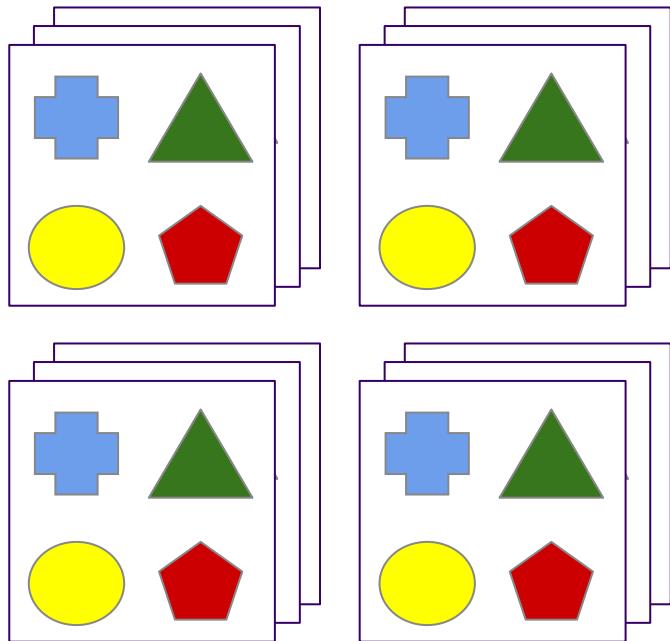


A microservices architecture puts each element of functionality into a separate process

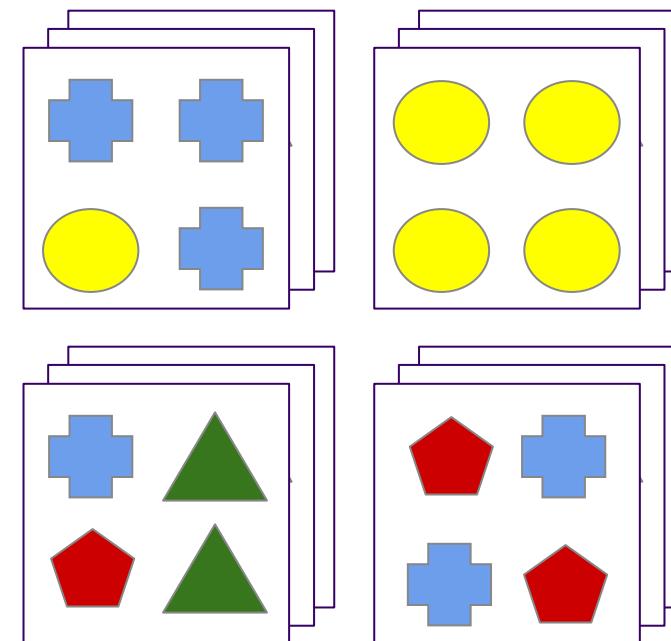


Monolith vs Microservices

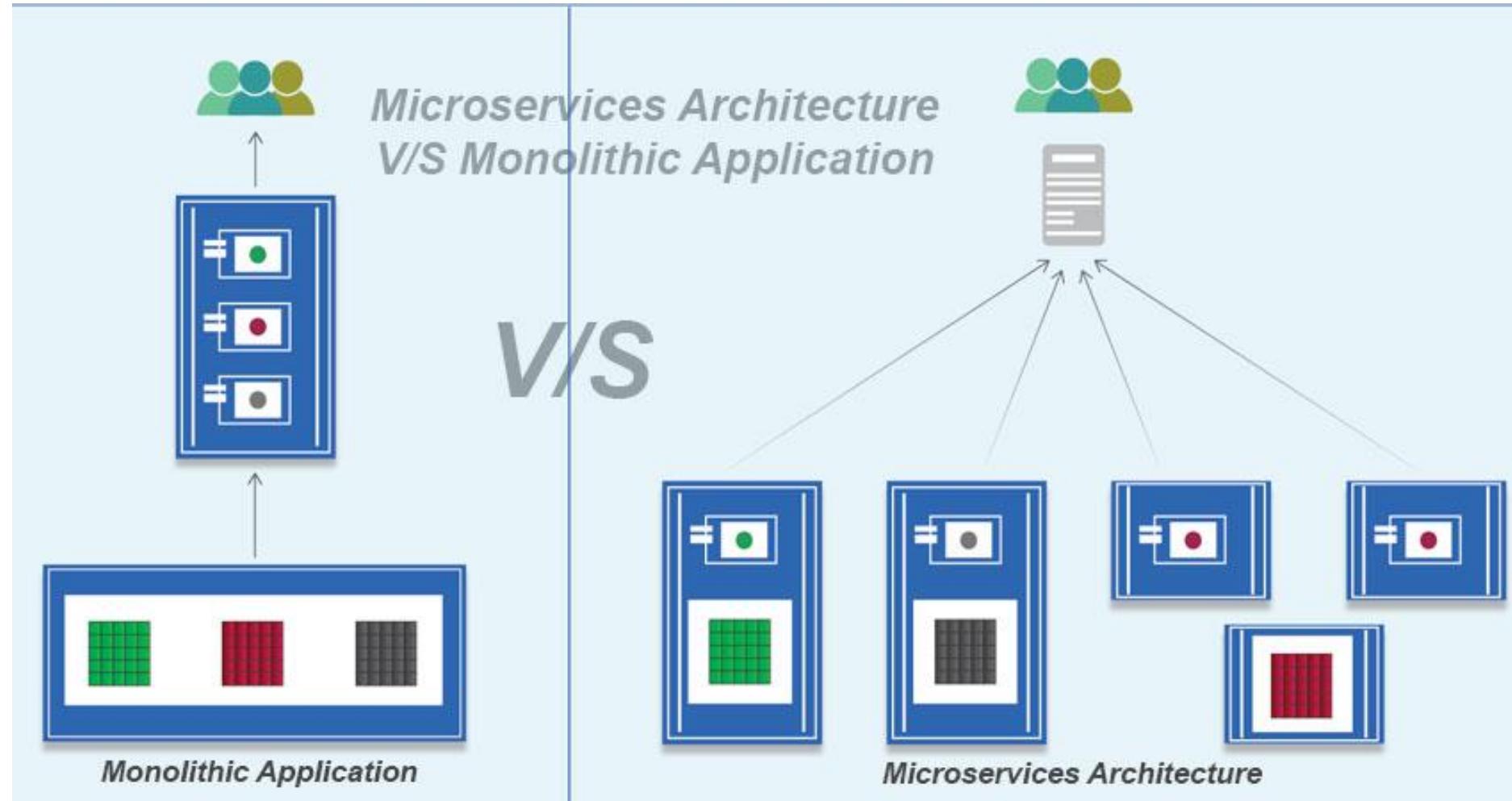
Monolith scales by replicating the whole thing on multiple servers



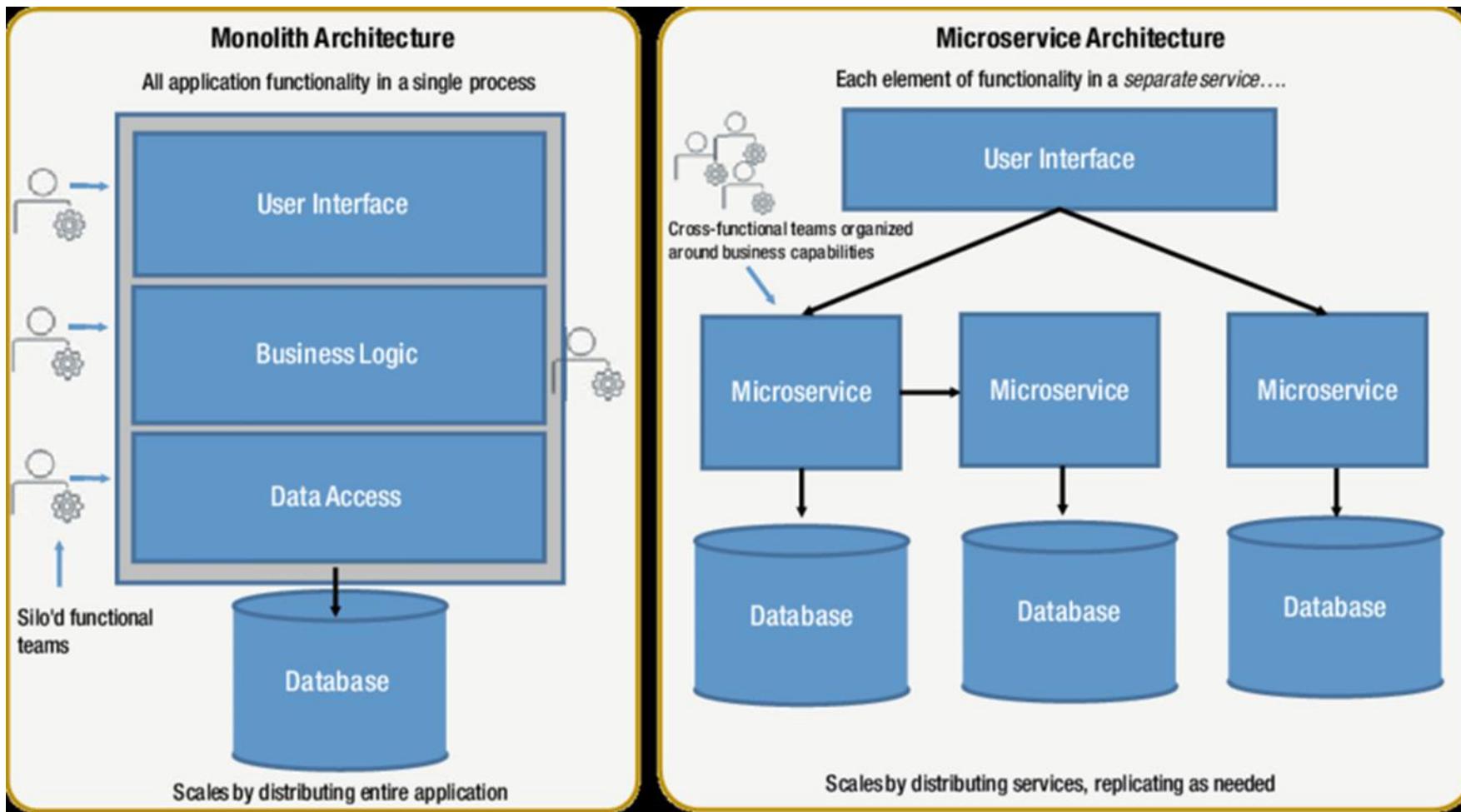
Microservices scales by distributing these services across servers, replicating as needed.



Monolith vs Microservices



Monolith vs Microservices



04. Characteristics of Microservices Architecture

Characteristics

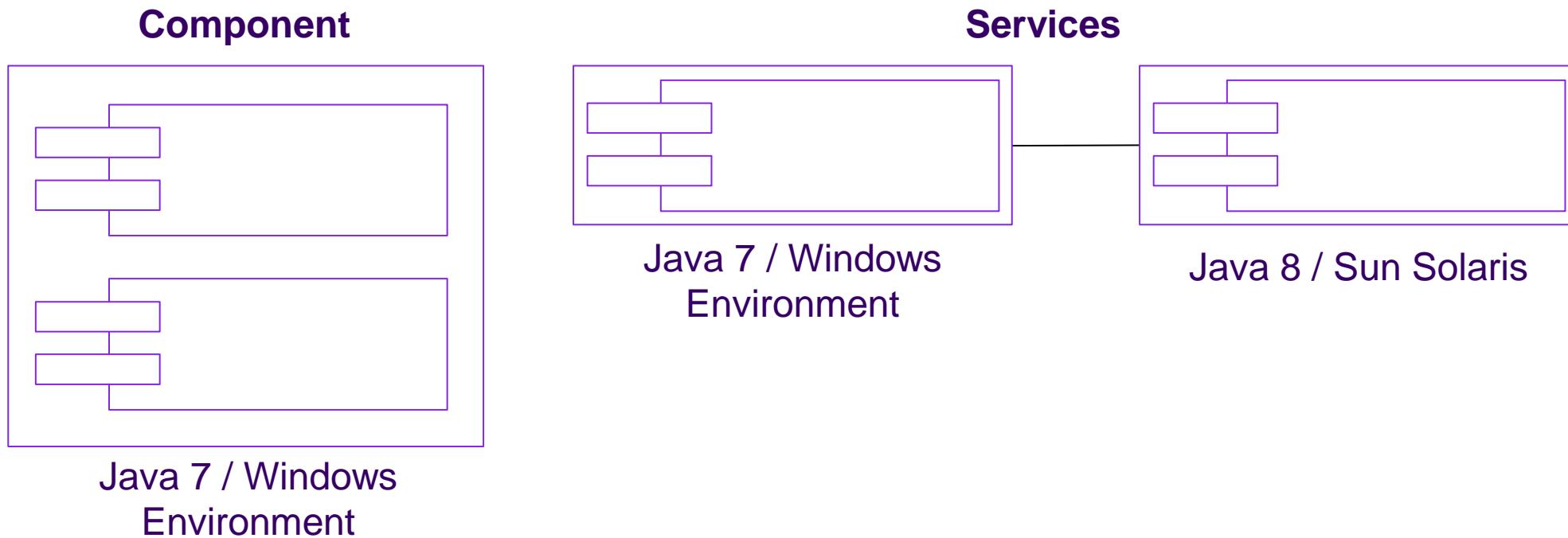
Microservices Architecture

- Componentization via services
- Organized around business capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure

Characteristics

Componentization via Services

- Component is a unit of software that is independently replaceable and upgradeable.
- Component can be a Library or a Service.



Characteristics

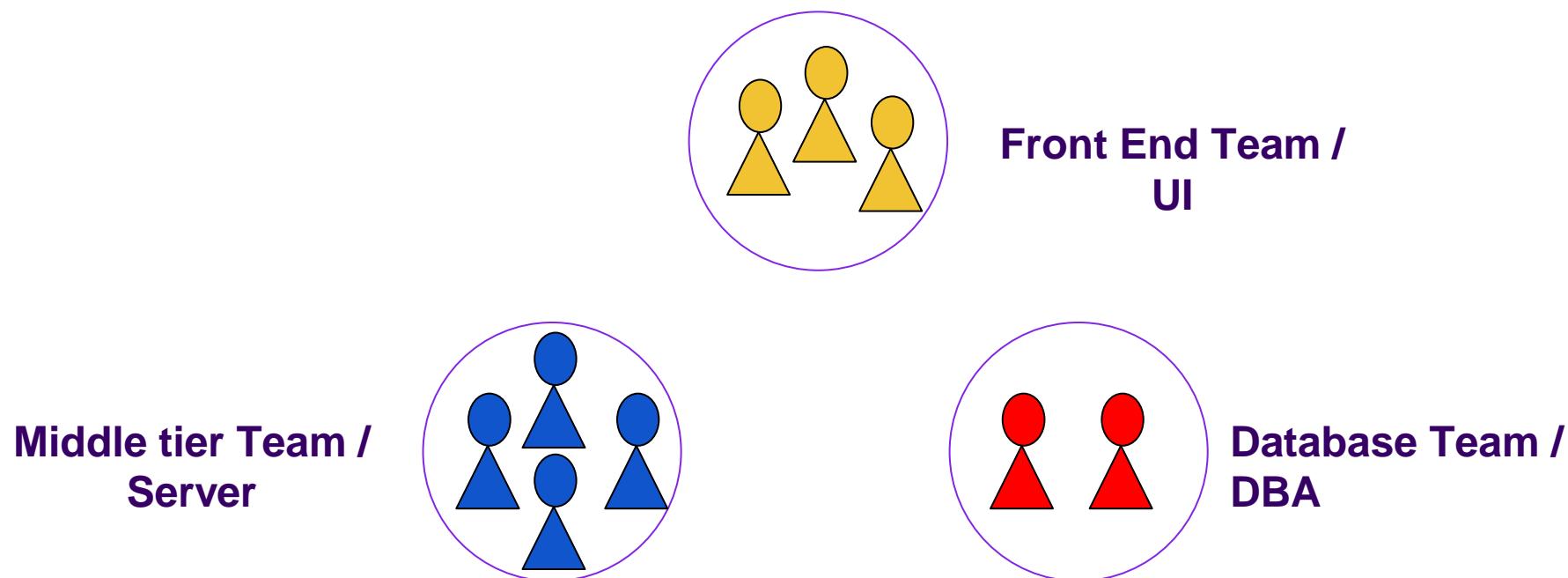
Componentization via Services

- If the component is a library, it is difficult to upgrade it since it affects to the whole application.
Ex: Upgrading one component to a newer version of Java needs the whole application to be upgraded
- Communication between library components introduce high coupling.
- If the components are services, they could be running in different runtimes, and you may use web services or messaging queues to communicate with them.
- Those services can be independently replaceable and upgradeable.

Organized Around

Business Capabilities

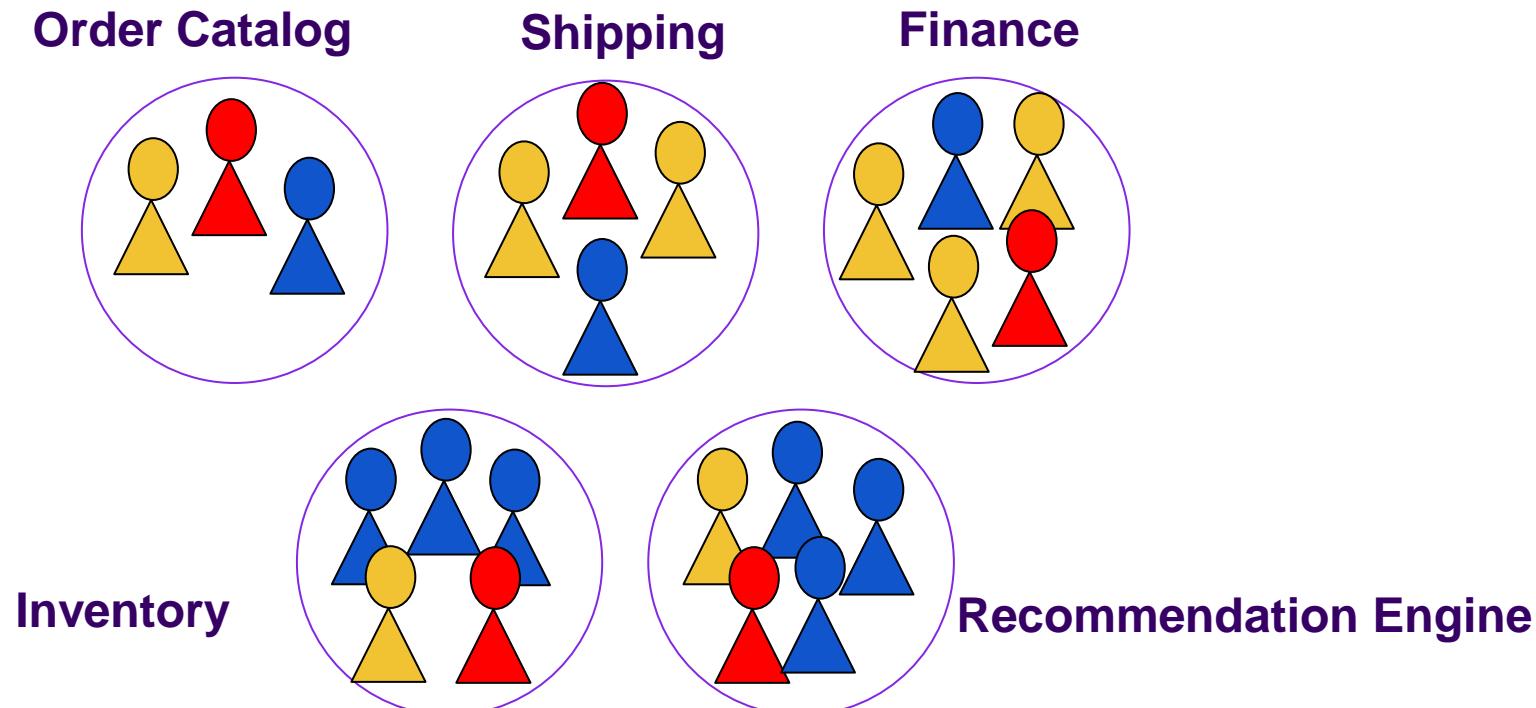
- Most of the software development organizations organize the team according to the technology.



Organized Around

Business Capabilities

- But when implementing a solution according to the Microservices approach, you should organize the team according to the business capabilities.



Organized Around

Business Capabilities

- These teams are cross functional, including full range of skills in all aspects of development, user experience, project management, ...etc.
- How big a particular team should be?
It depends on the business capability and usually it should be no more than a 2 pizza team (About 24 people)
- Each team is responsible for all aspect of software development including the communication right up to the end user experience.

Characteristics

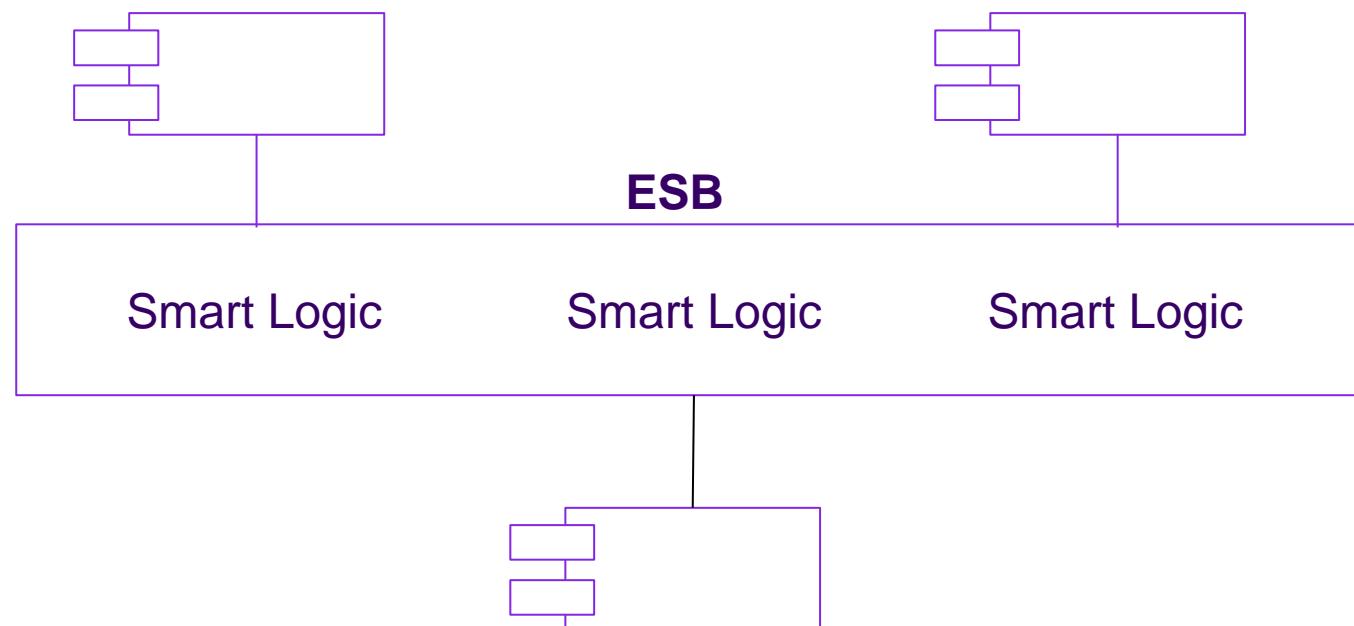
Products not Projects

- In traditional project model the ultimate goal is to deliver a piece of software which is then considered as completed.
- After that it is handed over to the maintenance and support organization.
- But in Microservices approach the team who developed the component owns it over its full lifetime.
- Hence the components to be built in Microservices approach is considered as products.
- Product team is responsible for monitor it in production, its availability, new enhancements, bug fixing, end user experience and some of the support work.

Characteristics

Smart Endpoints and Dumb Pipes

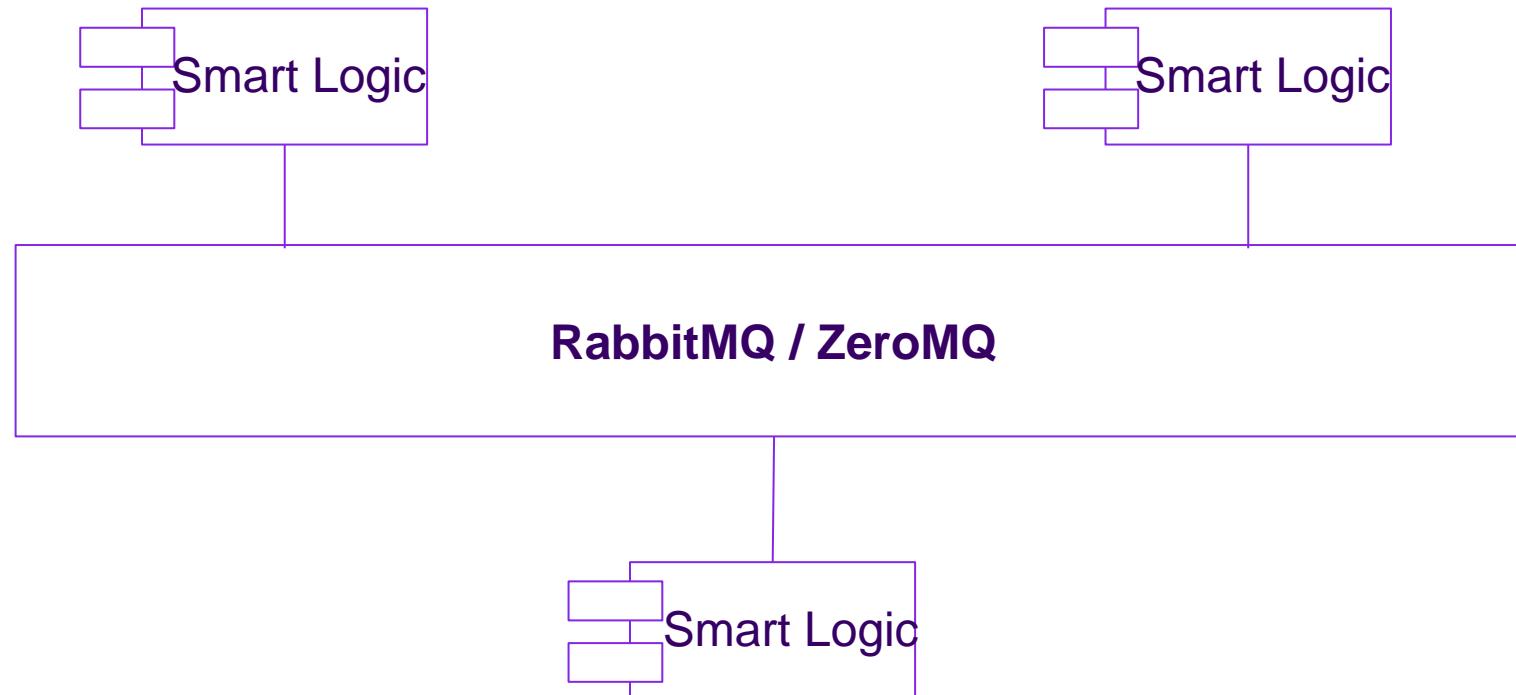
- ESB is a powerful piece of middleware used in SOA which can route messages, apply business rules, message transformations, ...etc.
- So, most of the intelligent logic (smarts) is written in the ESB itself.



Characteristics

Smart Endpoints and Dumb Pipes

- But Microservices approach rejects this notion and moves the smarts to the endpoints.



Characteristics

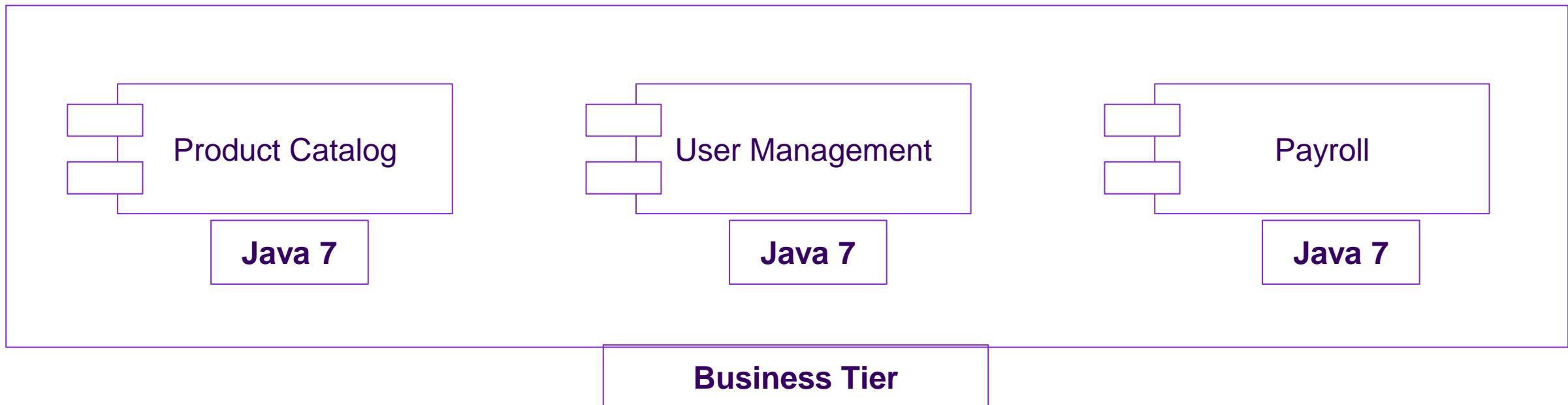
Smart Endpoints and Dumb Pipes

- The main reason for that is Microservices approach favor in building applications as decoupled and as cohesive as possible.
- So, they rely on intelligent endpoints and a dumb but efficient piping mechanism.
- The main inspiration for this is the internet itself which works very well because its a dumb set of pipes where the intelligence is implemented at the endpoints.

Characteristics

Decentralized Governance

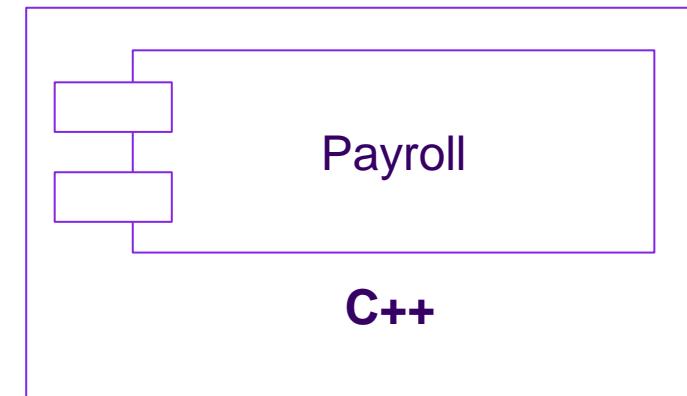
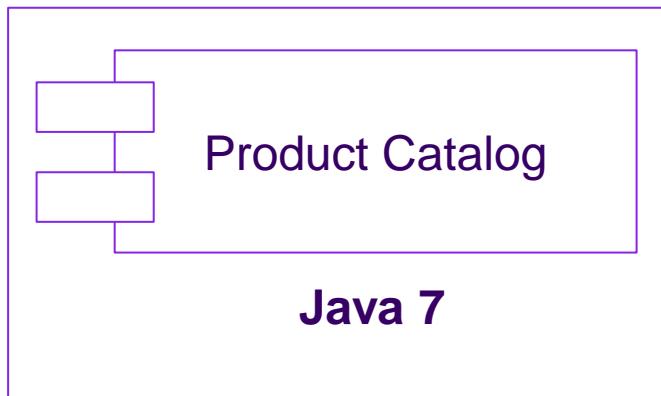
- Most of the Monolithic approaches employ a centralized governance where there are strict set of rules regarding the technologies and patterns to be followed.
- At least all modules of the application should be written in same language.



Characteristics

Decentralized Governance

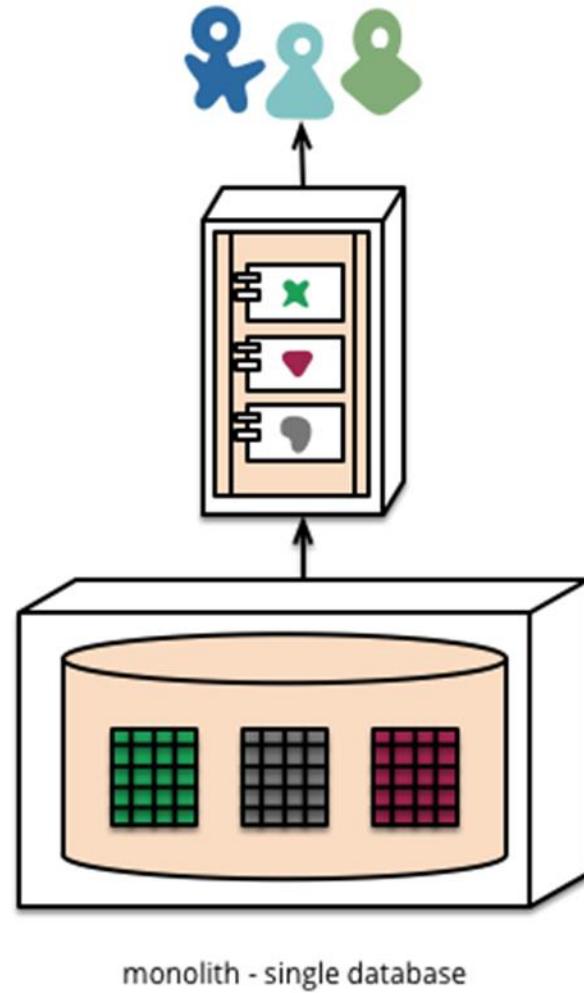
- Since Microservices approach has separate services where each running in its own process, the product team can decide on which language to write the components and what tools and technologies to use.



Characteristics

Decentralized Data Management

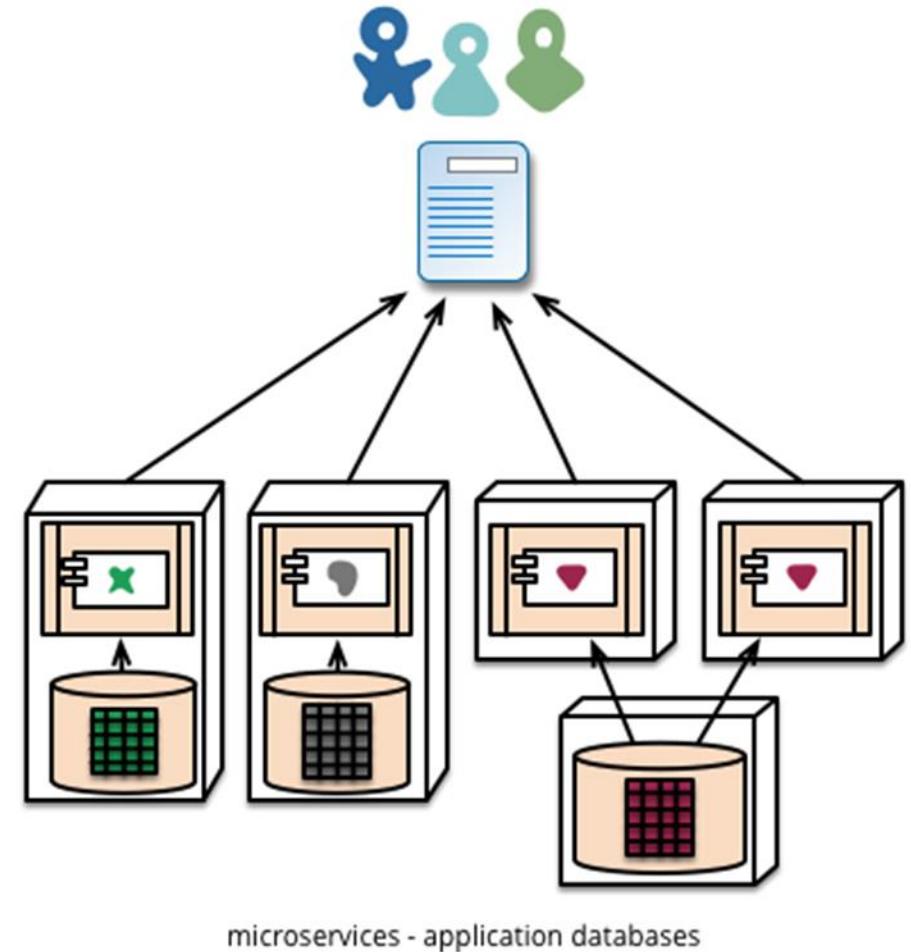
- In Monolithic approach, all data persist in one database.
- Any upgrade to the database will affect all business components of the application.



Characteristics

Decentralized Data Management

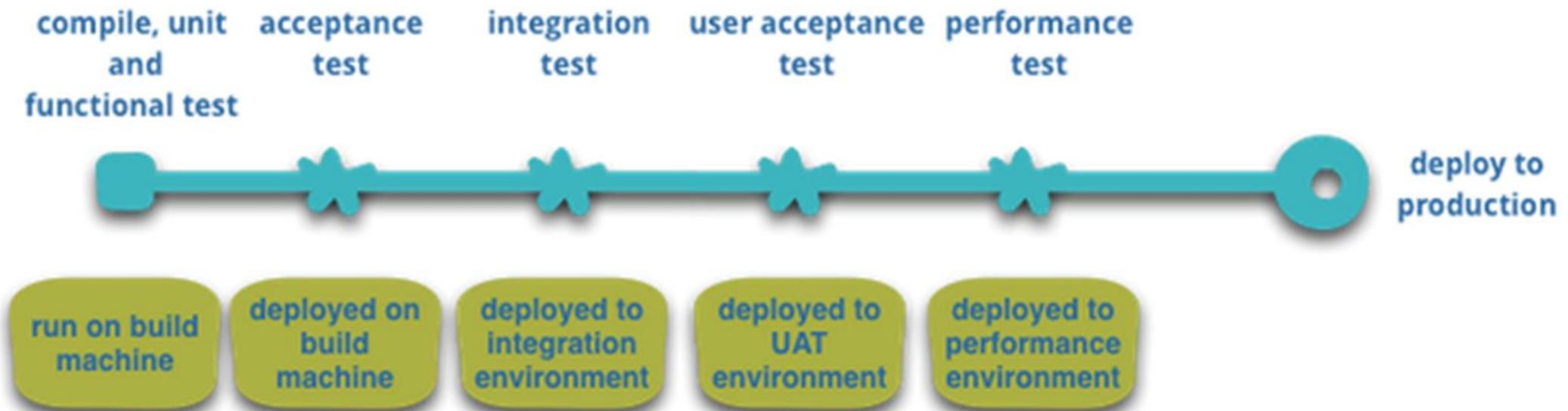
- Using Microservices approach, every service is responsible for its own data and own persistence.
- However, a particular service cannot talk to another services data store directly.
- Communication should always occur through the other service's API.
- This provides the freedom for each product team to decide on which datastore to choose according to the business capability they are implementing.



Characteristics

Infrastructure Automation

- In order to make everything working smoothly in Microservices, infrastructure automation has become an absolute necessity.



Characteristics

Infrastructure Automation

- Some of the features such as continuous delivery and blue green deployment has become mandatory if you are choosing the Microservices approach.
- Since it involves several services running with zero downtime and ensuring the consistency throughout the application, monitoring has also become an absolute necessity too.

Characteristics

Design for Failure

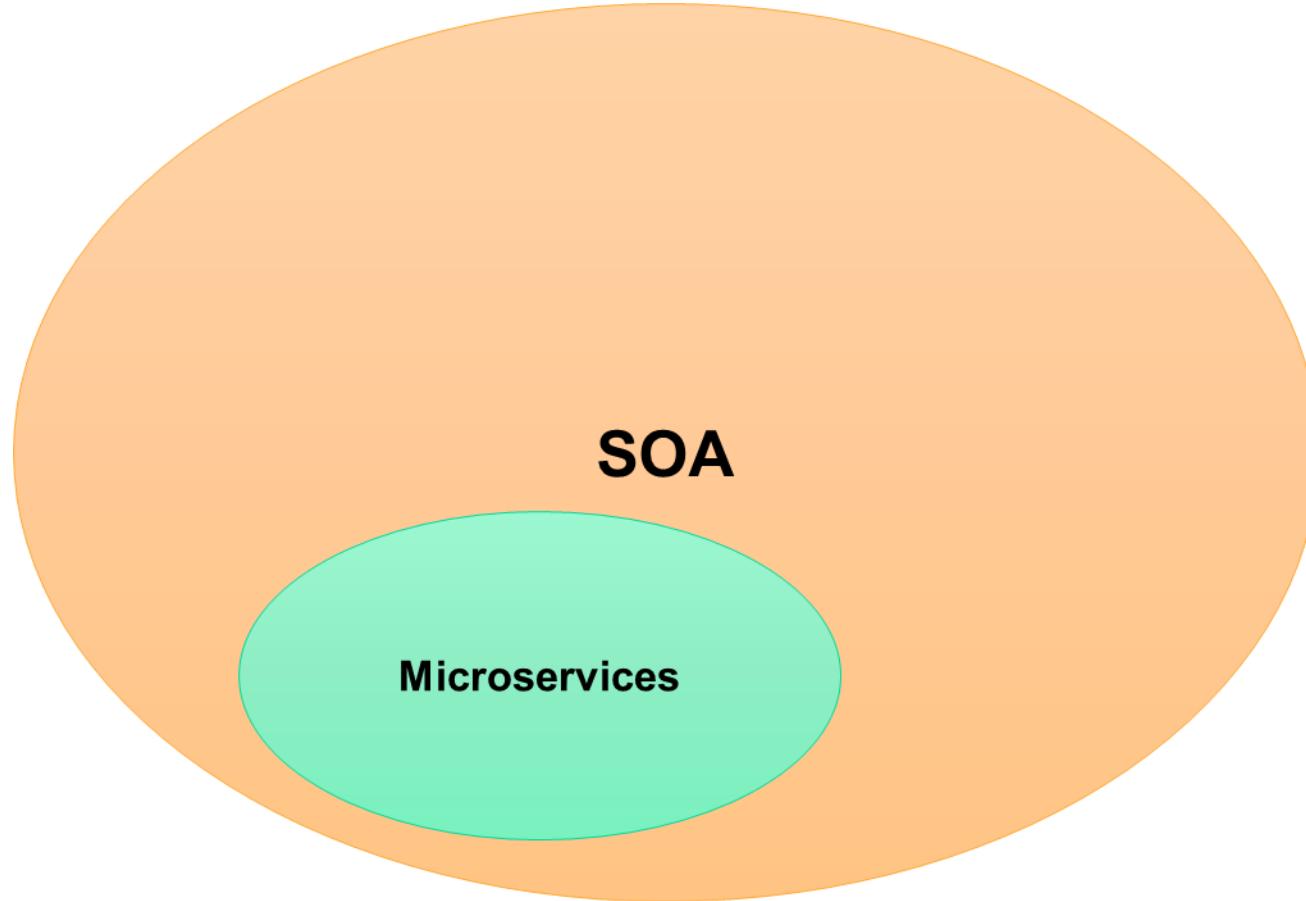
- Any service may fail in a given time in Microservices approach.
- Since all services communicate with each other via their APIs, failure in one service may affect the business flow of another service as well.
- Therefore, it is utmost important to have certain tools to monitor the health of the services so that when a service is down, the product team can work on it as soon as possible.
- In addition to that Microservices approach use certain tools to randomly making the services down in order to test how resilient their application is and how quickly they can recover the damage.

Ex: Chaos Monkey by Netflix

05. Microservices vs SOA

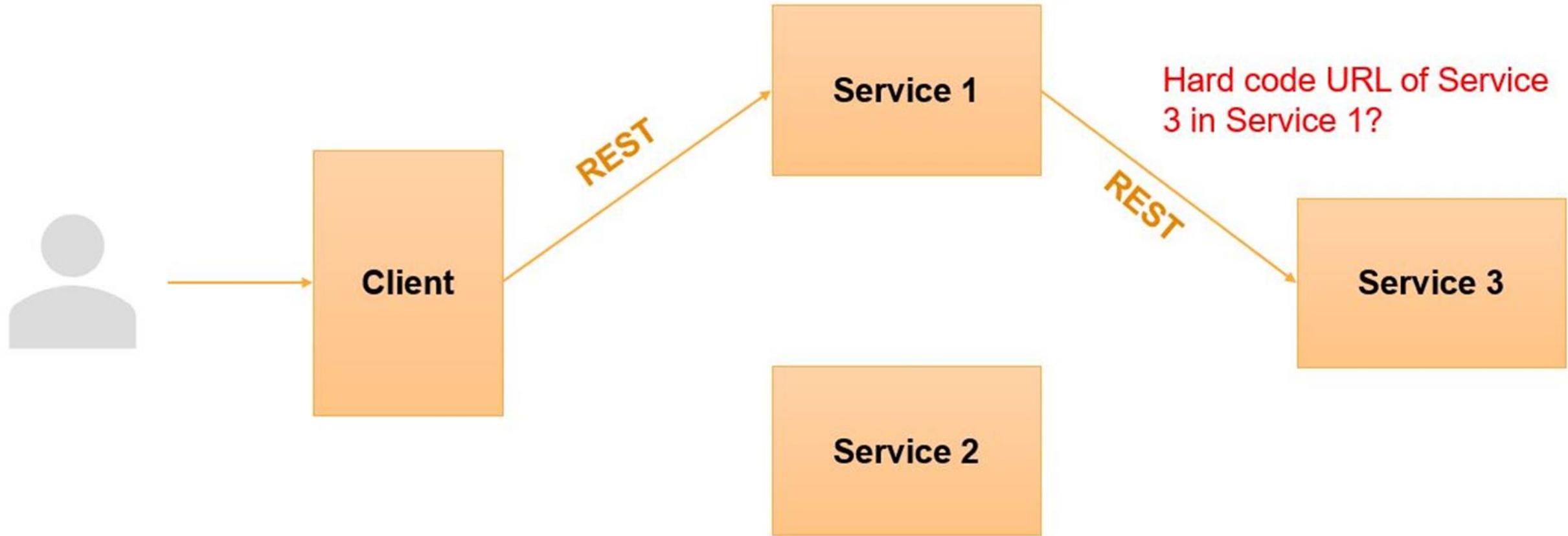
Microservices vs SOA

- Are microservices just SOA?



06. Communication Between Microservices

Communication Between Microservices

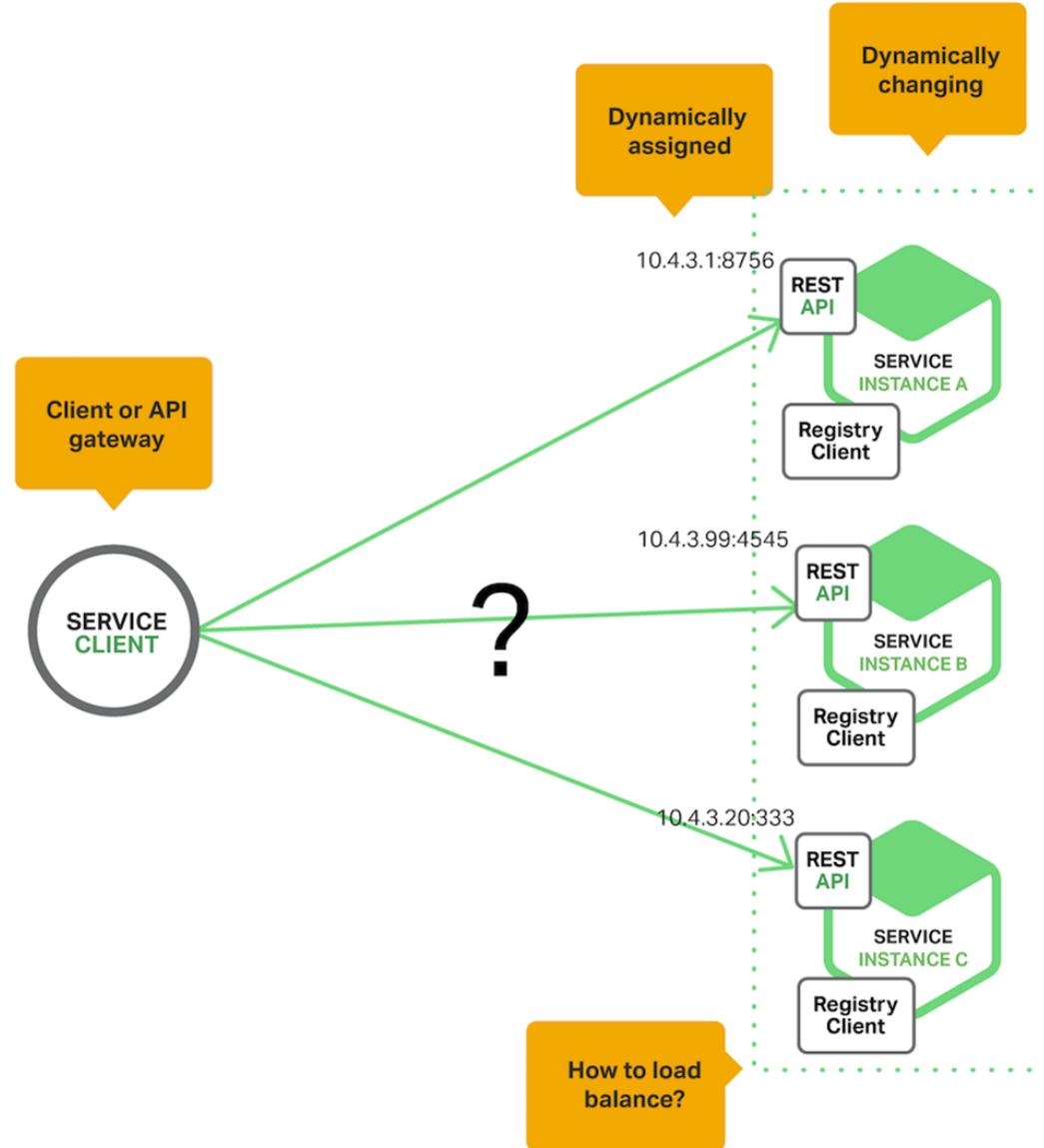


Communication Between Microservices

Why hard coding of URLs in Microservices are bad?

- Changes in URL of a microservice will require code updates in others.
- Dynamic URLs in the cloud.
- Load balancing
- Multiple environments (DEV, TEST, PROD)

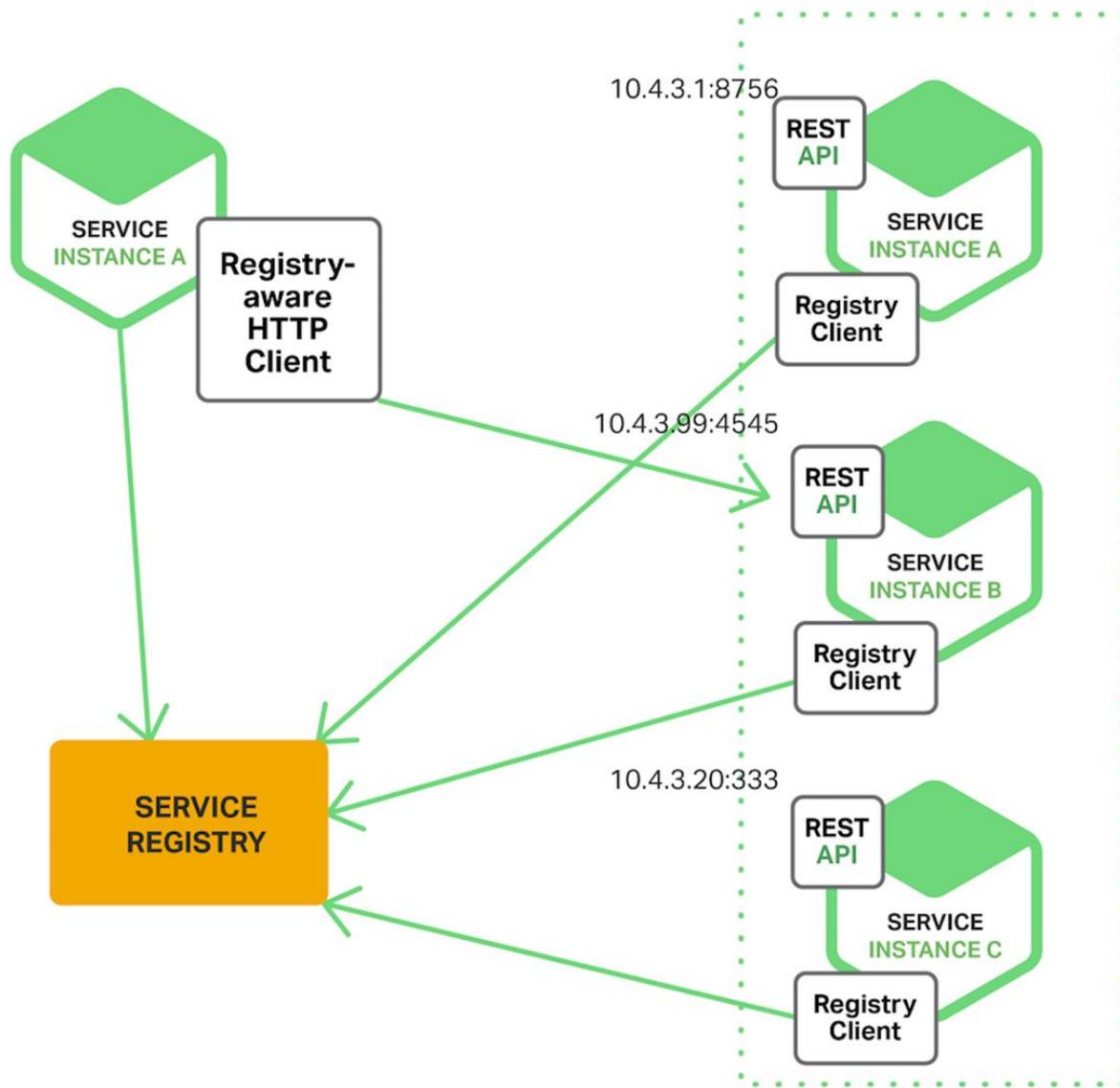
Communication Between Microservices



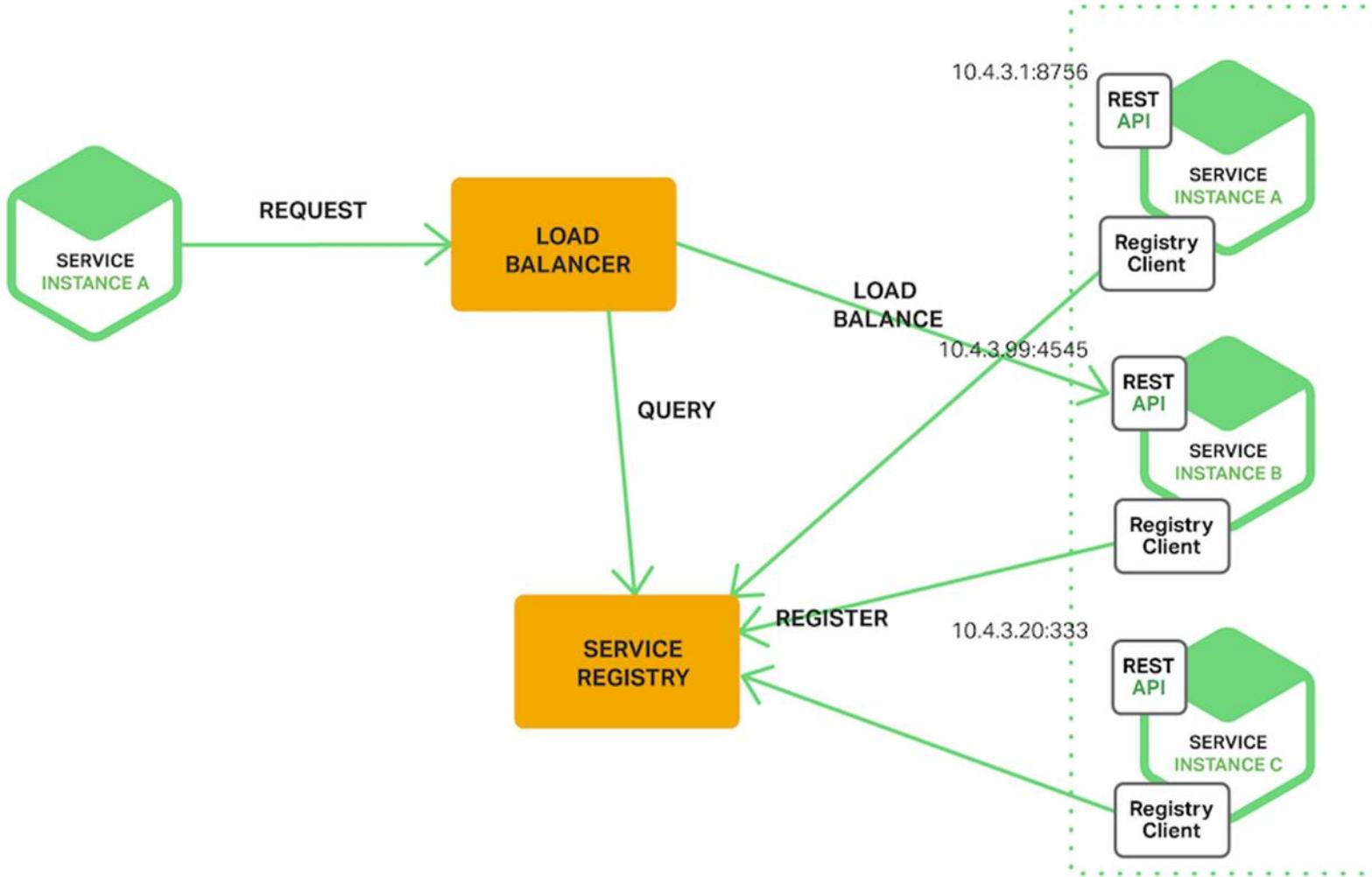
Service Discovery in Microservices

- Automatically registers microservices in a service registry.
- This is itself a microservice and we call it the discovery server.
- All microservices can refer the service registry when they need to talk to another microservice.

Client-Side Service Discovery



Server-Side Service Discovery



07. Advantages & Disadvantages of Microservices

Advantages of Microservices

Partial deployment

- You can upgrade services without affecting others.

High availability

- Even though your recommendation engine is shut down, you can still use the shopping cart functionality.

High scalability

- Demanding services can be deployed in multiple servers

Preserve modularity

- Business capabilities are separated, and coupling is low because of following the single responsibility principle

Freedom to choose technology

- Best possible technology can be used to write a specific business capability.

Advantages of Microservices

Faster deployments

Easier to understand the codebase

- Microservice focuses on one business capability.

Improves fault isolation

- Application may remain largely unaffected by the failure of a single module.

Disadvantages of Microservices

- Difficult to implement multiple database transaction logic
- Testing a Microservices based application can be cumbersome.
- Poor consistency
- Complex to understand the whole architecture of the application.
- Redundant code (classes) may span across multiple Microservices.

A young woman with long brown hair, wearing an orange sweatshirt and a tan backpack, stands in a classroom. She is holding several books, including a yellow spiral-bound notebook and a red book. She is looking upwards and to the right with a thoughtful expression. The background shows desks, chairs, and a chalkboard with some writing on it.

Wrap-up

Thank you!

#MOMENTOFSERVICE





© COPYRIGHT© 2021 BY INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB (PUBL). ALL RIGHTS RESERVED. THIS MATERIAL AND ITS CONTENT IS PRODUCED BY THE IFS ACADEMY FOR AUTHORIZED TRAINING PURPOSES ONLY AND REMAINS THE INTELLECTUAL PROPERTY OF IFS. NEITHER THE MATERIAL OR ITS CONTENT MAY BE COPIED, REPRODUCED, OR DISTRIBUTED WITHOUT IFS' EXPRESS WRITTEN PERMISSION.

IFS DOES NOT WARRANT, EITHER EXPRESSLY OR IMPLIED, THE ACCURACY, TIMELINESS, OR APPROPRIATENESS OF THE INFORMATION CONTAINED IN THIS TRAINING MATERIAL AND DISCLAIMS ANY RESPONSIBILITY FOR CONTENT ERRORS, OMISSIONS, OR INFRINGING MATERIAL. IFS ALSO DISCLAIMS ANY RESPONSIBILITY ASSOCIATED WITH RELYING ON THE INFORMATION PROVIDED IN THIS DOCUMENT AND ANY AND ALL LIABILITY FOR ANY MATERIAL CONTAINED ON OTHER CHANNELS THAT MAY BE LINKED TO THE IFS TRAINING MATERIAL.