Lecture Intro (1)

Introduction to Cloud Computing:

* Cloud computing provides ubiquitous (=everywhere), convenient and on demand access to shared pools or configurable computing resources such as servers, storage, applications and services.
* Key attributes include scalability, rapid provisioning and reduced management effort.

Traditional vs Cloud Computing Models

* Traditional model
  + Organizations owned or leased hardware and software resources
  + Significant capital expenditure (CAPEX) and risks
  + Challenges included overprovisioning and underutilization
* Cloud Model
  + IT resources provided as a service
  + Pay as you go model reduces CAPEX and shifts focus to operational expenditure (OPEX)

Cloud Computing Service Models

* Laa5 (Infrastructure as a Service: Provision of hardware resources like VMs, storage and network.
* PaaS (Platform as a Service): Tools for application development.
* SaaS (Software as a Service): Ready to use applications hosted in the cloud
* Other Services: Security, AI, databases and analytics as services

Principles of Cloud Computing

* Resource Pooling: Aggregation of resources to serve multiple users dynamically.
* Multitenancy: Sharing physical infastructure among users while ensuring isolation
* Elasticity: Automatic scaling of resources based on demand.
* Pay as You go: Payment based on usage
* Automation: Dynamic provisioning/deprovisioning of resources through APIs

Benefits of Cloud Computing

* Economic
  + Reduction in CAPEX and operational costs
  + Economies of scale and global reach
* Productivity
  + Faster time to market and competitive advantage
  + Simplified infrastructure management
* Other Benefits
  + Enhanced performance, reliability, availability and security.
  + Agility in software development via services like data storage, monitoring and alerting.

Cloud vs Grid Computing

* Grid computing
  + Distributed systems leveraging multiple geographically distributed computers.
  + Suited for large-scale scientific computations
* Cloud Computing
  + Centralized resource management and services through virtualization

Suggested Questions for Exam Preparation for this Lecture

1. Define Cloud Computing and explain its key characteristics
2. What are the three main cloud computing service models?
3. Explain the concept of resource pooling and multitenancy in cloud computing
4. How does the pay as you go model benefit organizations economically?
5. List the primary benefits of cloud computing and explain how it enhances productivity and reduces costs
6. Describe the role of virtualization in enabling cloud computing
7. Differentiate between grid computing and cloud computing with examples.

Model Answers of the above questions

1. Cloud computing is a model for enabling ubiquitous, on-demand network access to shared pools of configurable computing resources such as servers, storage, applications, and services. These resources can be rapidly provisioned and released with minimal management effort or service provider interaction.   
     
   Key Characteristics:  
   On demand Self Service: Users can provision resources as needed automatically   
     
   Broad Network Access: Resources are accessible over the internet from various devices  
     
   Resource Pooling: Resources are shared among multiple users while maintaining isolation  
     
   Elasticity: Automatic scaling of resources based on demand  
     
   Measured Service: Pay as you go pricing model
2. There are three PaaS, SaaS and last laaS
3. Resource Pooling is:   
     
   Aggregating computing resources (e.g., servers, storage, and networks) to serve multiple users dynamically. Resources are allocated and reallocated based on demand using techniques like load balancing and predictive analytics.  
     
   Multitenancy:  
     
   Multiple users (tenants) share the same physical infrastructure while remaining isolated from each other. This ensures that one user’s activities do not impact others. Virtualization enables multitenancy by partitioning hardware into separate virtual machines.
4. The pay-as-you-go model ensures organizations only pay for the resources they use. This eliminates overprovisioning and underutilization costs, reduces upfront CAPEX, and aligns costs directly with business growth. It is particularly beneficial for startups and businesses with fluctuating demands.
5. **Cost Savings**: Pay-as-you-go pricing eliminates CAPEX and reduces OPEX.  
     
   **Productivity**: Reduced time to market and faster development cycles due to pre-built tools and services.  
     
   **Flexibility and Agility**: Resources can be scaled instantly to meet changing demands  
     
   **Global Reach**: Services can be accessed worldwide, reducing latency and labor costs.  
     
   **Security Improvements**: Providers offer advanced physical and network security measures.  
     
   **Performance and Reliability**: High availability and fault tolerance due to resource pooling and redundancy.
6. Virtualization creates virtual versions of hardware resources, allowing multiple virtual machines to run on a single physical machine. This enables resource pooling, multitenancy, and scalability. Virtualization also supports automation by enabling resources to be provisioned/deprovisioned dynamically.
7. Grid Computing:  
     
   Focuses on distributing computing tasks across geographically dispersed systems.  
     
   Used for large-scale scientific computations  
     
   Cloud Computing:  
     
   Centralized management of virtualized resources delivered as services over the internet   
     
   Examples include Amazon Web Services and Google Cloud Platform

Articles that he gave in the intro

Cloud Migration Framework for SMEs (Small and Medium Enterprises)

* SMEs often hesitate to adopt cloud computing due to lack of expertise, concerns over costs and data privacy issues.
* Migration frameworks include pre migration assessments, feasibility checks and service mapping to optimize cloud adoption
* Key considerations for SMEs:
  + Interoperability: understanding how on premises services fit cloud models
  + Provider Selection Criteria: Based on service offerings, costs, compliance and support
  + Data Privacy and Security: Often mitigated by shared responsibility models offered by providers

Economic Impacts of Cloud Computing

* Cloud computing offers significant cost saving through reduced CAPEX and operational efficiency, particularly for SMEs
* Adoption boosts economic growth by creating jobs, fostering innovation and enabling business to scale efficiently.

Case Study: Migration to IaaS (Infrastructure as a Service)

* Mitigating legacy systems to cloud computing like Amazon EC2 can reduce infrastructure costs by 37% over five years
* Risks include dependency on third party providers and initial learning curves for IT teams
* Support issues can be minimized through cloud hosted solutions

Benefits and Risks in Cloud Migration

* Benefits
  + Scalability
  + Cost Efficiency
  + Global Reach
  + Innovation
* Risks
  + Dependency on providers
  + Data security and compliance issues
  + Challenges in managing hybrid environments

WE ARE NOT OBLIGED TO LEARN THESE THINGS FROM THE ARTICLES BUT LEARN LIKE 1-2 QUESTIONS SO YOU KNOW WHAT IS GOING ON

Lecture 2 (VIRTUALIZATION)

Virtualization Types

* Full Virtualization
  + Provides complete hardware emulation using a hypervisor (Type-1 or Type2)
  + Binary translation ensures isolation by trapping critical instructions and translating them
  + Advantage : Isolation and compatibility with unmodified guest OS
  + Disadvantage : Performance penalty due to binary translation
* Host-Based Virtualization (Type-2)
  + Hypervisor runs as an application on a host OS
  + System calls from guest OS are passed through the host OS to real hardware
  + Advantage : Simplified deployment and compatibility with diverse guest Oss’s
  + Disadvantage : Performance overhead due to dual layered interaction
* Paravirtualization
  + Guest OS is modified to interact directly with the hypervisor using hypercalls
  + Advantage : Better performance by avoiding binary translation
  + Disadvantage : Requires source level OS modification, reducing portability

Hypervisors

* Type-1 (Bare Metal)
  + Directly installed on hardware, offering better performance and isolation
  + Used for server environments requiring high performance
* Type-2 (Hosted)
  + Runs on top of a host OS, suitable for desktop virtualization scenarios

Advanced Virtualization Techniques

* Binary Translation
  + Converts privileged instructions into safe instructions executed by the hypervisor
  + Introduces performance penalties but ensures security and isolation
* Hardware-Assisted Virtualization
  + Introduced by Intel (VT-x) and AMD (AMD-V) to reduce software overhead
  + Shadow registers and extended page tables optimize translation and reduce hypervisor interventions
* Virtual machine control Structure (VMCS)
  + Manages transitions between host and guest states efficiently
  + Reduces latency during VM exits and entries
* SR-IOV (Single Root I/O Virtualization)
  + Provides virtual functions (VFs) for each VM, allowing direct access to I/O hardware, reducing hypervisor intervention

Virtualization Tools

* Xen
  + Paravirtualization hypervisor developed by the University of Cambridge
  + Features include User domains and Dom0 (which manages hardware access)
* Emulation
  + Simulates hardware of a completely different architecture (emulators)
  + High overhead but enables software compatibility

Suggested Questions for the Above Lecture

1. Define Virtualization and describe the main types of virtualization
2. Explain the role of hypervisors and differentiate between Type-1 and Type-2 hypervisors
3. What is binary translation and why does it impact virtualization performance
4. Compare full virtualization, paravirtualization and hardware assisted virtualization
5. How do shadow registers and extended page tables improve virtualization performance
6. Explain the role of VMC’s in hardware assisted virtualization
7. What are they key differences between emulation and virtualization
8. Discuss the security implications of virtualization focusing on guest isolation and hypervisor responsibilities

Model Answers

1. Virtualization is the process of creating a virtual version of hardware or software resources, such as servers, storage, or networks. It enables multiple operating systems (OSs) to run on a single physical machine while maintaining isolation and efficient resource utilization.  
     
   Main Types: Full Virtualization, Paravirtualization, Host Based Virtualization, Hardware-Assisted Virtualization.  
     
   FV: Emulates complete hardware; guest OS runs unmodified  
   PV: Guest : Guest OS is modified to interact with the hypervisor using hypercalls  
   HBV: The hypervisor runs on top of a host OS  
   HAV: Uses CPU features like Intel VT-x or AMD-V to reduce overhead
2. A hypervisor is software that creates and manages virtual machines providing isolation and resource allocation between guest OS’s  
     
   Type-1: Installed directly on hardware, offering high performance and low latency  
   Type-2: Runes on a host OS, suitable for desktops  
     
   Key difference: Type-1 interacts directly with hardware, while Type-2 depends on the host OS
3. Binary translation is a technique where the hypervisor intercepts and translates privileged instructions issued by the guest OS into safe instructions that can be executed on the host hardware   
     
   Performance Impact:  
   a)Increases overhead because each critical instruction triggers multiple translation   
   b)Slower compared to direct hardware execution, but ensures guest isolation and security
4. Full Virtualization  
   Emulates complete hardware using a hypervisor   
   Pros: No OS modification needed supports closed sources OS’s  
   Cons: Slower due to binary translation  
     
   Paravirtualization  
   Guest OS modified to communicate with the hypervisor via hypercalls  
   Pros: Better performance avoids binary translations   
   Cons: Requires OS modification; limited portability   
     
   Hardware Assisted Virtualization  
   Utilizes CPU extensions like VT-x and AMD-V to handle privileged instructions directly  
   Pros: Reduced overhead, supports unmodified OS  
   Cons: Requires compatible hardware
5. Shadow Registers : Maintain duplicate copies of critical CPU registers. Guest OS changes are recoded in shadow registers without affecting real hardware reducing hypervisor intervention  
     
   Extended Page Tales: Eliminate two level address translation by mapping guest virtual addresses directly to host physical addresses, reducing memory access latency and improving efficiency
6. The virtual machine structure manages the state transitions between the hypervisor and guest OS’s  
   Functions:  
   a) Stores guest and host states during VM exits and entries   
   b) Enables efficient switching by minimizing memory accesses for state management  
   c) Improves performance by leveraging specialized CPU registers for VM control
7. Emulation: Simulates an entirely different hardware architecture  
   Pros: Broad compatibility  
   Cons: Significant performance overhead  
     
   Virtualization: Creates virtual versions of hardware on the same architecture  
   Pros: Near- native performance, efficient resource use  
   Cons: Limited to compatible hardware/software
8. Security Implications:  
   a) Guest Isolation: Ensures that VMs cannot directly interfere with each other by sandboxing resources  
   b) Hypervisor Role: Acts as a security boundary, managing resource access and mitigating risks like VM escape   
   c) Risks: Dependence on hypervisor integrity and Shared Resources can introduce side channel vulnerabilities if not properly isolated

Lecture 3 (IaaS and Public Key Crypto Primer )

Definition and Overview of IaaS

* Infrastructure as Service (IaaS)
  + Provides virtualized computing resources over the internet
  + Key components: Virtual machines, storage, networking and load balancers
  + Users have the greatest control over infrastructure compared to other cloud models

Key Features of IaaS

* Elasticity: Resources are provisioned and deprovisioned dynamically based on demand
* Cost Model: Pay-as-you-go metered usage in increments like hours or seconds
* Multi-Tenancy: Allows sharing of resources among multiple users, improving cost efficiency

Amazon Web Services Example (AWS)

* EC2 instances:
  + Virtual servers created from Amazon Machine Images
  + Users can choose OS, storage and configurations
* Elastic Block Store
  + Persistent block storage ideal for databases and high performance workloads
* Simple Storage Service
  + Object storage for large scale data analytics and backups
  + Features like versioning replication and lifecycle management
* Elastic File System
  + Shared scalable file storage for concurrent access across multiple instances

Benefits and Challenges (to using IaaS)

* Benefits
  + Cost-saving by eliminating upfront hardware investments
  + High scalability and availability for workloads
  + Disaster recovery and business continuity support
* Challenges
  + Security: Protecting data in multi tenant environments
  + Vendor Lock in: Difficulty migrating between providers due to proprietary configurations

Public Key Cryptography

Cryptography Basics

* Symmetric Cryptography
  + Uses a single shared secret key for both encryption and decryption
  + Efficient but less secure for key exchange
* Asymmetric Cryptography (Public key Cryptography)
  + Uses a key pair: a public key encryption and a private key for decryption
  + Enables secure communication without sharing a private key

Public Key Cryptography

* Public keys can be shared openly
* Ensures secure data exchange, but less efficient than symmetric methods
* Often combined with symmetric encryption for better performance

Digital Certificates and Trust

* Digital Certificates
  + Issued by trusted Certification Authorities to authenticate public keys
  + X.509 protocol establishes a hierarchical trust model
* Hashing for Integrity
  + Algorithms like SHA-256 generate unique message digests to verify data integrity

Suggested Questions for IaaS

1. What is IaaS and how does it differ from PaaS and SaaS
2. Explain the pay as you go model in IaaS and its advantages
3. Discuss the role AWS EC2 and EBS and IaaS
4. What are the primary benefits of IaaS for businesses
5. Explain the challenges of security and vendor lock-in in IaaS

Suggested Questions for Public Key Cryptography

1. Differentiate between symmetric and asymmetric cryptography
2. Explain the concept of public key cryptography and its significance
3. What are the digital certificates and how do they ensure trust in public key cryptography
4. Why is asymmetric cryptography often combined with symmetric encryption
5. What role does hashing play in data integrity

Models Answers for IaaS

1. Infrastructure as Service provides virtualized computing resources such as virtual machines storage and networks over the internet  
     
   Differences:  
   a)IaaS: offers basic infrastructure. Users manage OS(AWS, Google Compute Engine), applications and runtime  
   b) PaaS: Provides tools for application development.(Heroku, Google App) Users focus on deploying applications   
   c)SaaS: Provides fully managed applications (Gmail, Microsoft 365). Users only consume the service
2. The pay as you go model charges users based on actual usage, such as CPU hours data transfer or storage space.  
     
   Advantages:  
   a)Cost efficient: Users avoid upfront capital expenses  
   b)Flexibility: Pay only for resources consumed reducing waste  
   c)Scalability: Automatically adjusts costs as usage scales up or down
3. AWS EC2: Provides virtual servers (instances) that can run various operating systems. Users choose configurations and manage applications.  
     
   EBS: Offers persistent block level storage for EC2 instances ideal for databases and workloads requiring high- performance storage.
4. Cost saving: Reduces hardware investments and maintenance costs  
   Scalability: Dynamically adjusts resources to meet business needs  
   Disaster Recovery: Ensures business continuity through redundancy and backups  
   Agility: Enables rapid deployment of infrastructure for new projects
5. Security: Multi tenancy may expose data to risks. Strong encryption, access controls and compliance are critical  
   Vendor Lock-in: Migrating between IaaS providers is complex due to proprietary configurations, APIs and tools. Business may incur significant costs during transitions

Model Answers for Public Key Cryptography

1. Symmetric Cryptography: Uses a single shared secret key for encryption and decryption. Efficient but requires secure key exchange   
     
   Asymmetric Cryptography: Uses a key Pair. The public key encrypts and the private key decrypts making it more secure but computationally expensive
2. Public key Cryptography uses two keys  
     
   Public Key: Shared openly to encrypt messages  
   Private Key: Kept secret to decrypt messages   
     
   Significance:  
   Enables secure communication without sharing private keys  
   Facilitates digital signature for message authentication and integrity
3. Digital certificates authenticate public keys  
     
   Issued by trusted Certification Authorities  
   Follow the X.509 standard for hierarchical trust.  
   A certificate verifies that a public key belongs to a specific entity ensuring trust and preventing impersonation
4. Asymmetric encryption is computationally expensive for large data. It is used to:  
   Secretly exchange a symmetric key   
     
   The symmetric key then encrypts and decrypts the bulk of data efficiently.  
   This combination provides both security and performance
5. Hashing ensures message integrity by generating a unique fingerprint for data  
     
   Algorithms like SHA-256 produce digests that change if the data is altered.  
     
   Verifying the hash ensures the message has not been tampered with during transmission.

Lecture 4 (PaaS and SOAs)

Definition and Overview of PaaS

Platform as a Service (PaaS)

* Provides a software framework for developing and deploying cloud applications.
* Includes development tools, runtime environments and the underlying infrastructure
* Example: Heroku offers support for multiple programming languages and simplifies the deployment of applications

Key Features

* Buildpacks
  + Simplify the build and deployment for applications
  + Two types: Official and Custom
* Dynos
  + Lightweight containers for running applications
  + Two types
    - Web Dynos: Handle web requests
    - Worker Dynos: Execute background tasks
* Add-ons
  + Modular tools that extend functionality such as message queues, databases and monitoring

Benefits

* Cost Saving: Reduces infrastructure costs by abstracting the hardware and system level management
* Agility: Accelerates development and deployment cycles
* Scalability: Dynamically adjusts resources to meet demand
* Focus on Innovation: Developers concentrate on application logic instead of infrastructure management

Challenges

* Security: Requires robust policies to secure data and access
* Vendor Lock in: applications tied to a specific platform may face migration difficulties

SOA

Definition and Overview of SOA

Service-Oriented Architecture (SOA):

* A design paradigm where applications are built as a collection of autonomous services
* Each service encapsulates specific business logic, accessible via API’s
* Promotes loose coupling, allowing services to be maintained and scaled independently

Key Features

* Autonomy
  + Services are self-contained with full control over their business logic
* Abstraction
  + Encapsulates internal working exposing functionality only through well defined interfaces
* Interoperability
  + Relies on open standards to ensure compatibility across different platforms
* Composability
  + Services can interact and combine into complex workflows

SOA Models

* Business services: Provide domain-specific logic
* Utility Services: Offer general-purpose functionality
* Controller Services: Orchestrate the interaction of business and utility services
* Enterprise service bus (ESB):
  + Facilitates communication between services
  + Provides routing, message mediation and centralized management

Benefits

* Scalability: Allows independent scaling of services
* Flexibility: Supports integration of legacy systems
* Discoverability: Services can be easily found and reused using registries

Challenges

* Complexity: Designing and managing SOA systems requires expertise
* Performance: Increased network overhead due to service communication

Suggested Questions for PaaS

1. What is PaaS and how does it differ from IaaS and SaaS
2. Explain the purpose and types of buildpacks in Heroku
3. What are the roles of web and worker dynos in PaaS
4. Discuss the benefits of using PaaS for application development
5. Explain the challenges associated with using PaaS, Particularly vendor lock-in

Suggested Questions for SOA

1. Define SOA and explain how it differs from traditional monolithic architectures
2. What is service abstraction and why is it important in SOA
3. Explain the role of an Enterprise Service Bus (ESB) in SOA
4. Discuss the benefits of SOA in terms of scalability and flexibility
5. What are the challenges associated with implementing SOA

Model answers for PaaS

1. Platform as a Service provides a cloud based environment for developing, testing, and deploying applications. It abstracts the infrastructure layer allowing developers to focus on writing code  
     
   Differences:  
     
   IaaS: Offers basic infrastructure like virtual machines, storage and networking  
   PaaS: Adds development tools and runtime environments to the infrastructure (Heroku)  
   SaaS: Provides fully managed applications for end users (Gmail)
2. Buildpacks automate the process of preparing and deploying applications on Heroku. They set up the appropriate runtime environment for the application   
     
   Types:  
   a)Official Buildpacks : Pre supported for popular languages like Python, Java and Node.js  
   b) Custom Buildpacks: Developed by users to cater to specific needs or non-standard applications
3. Web Dynos: Handle HTTP requests and serve web content. Typically include a web Server  
   Worker Dynos: Execute background tasks independently, such as database updates or batch jobs.
4. Cost Saving: Eliminates the need for hardware and system-level management  
   Agility: Accelerates development cycles and time to market  
   Scalability: Resources scale dynamically to handle variable workloads  
   Focus on Innovation: Developers can concentrate on coding rather than managing infrastructure
5. Vendor Lock-in: Applications developed for a specific PaaS platform may rely on proprietary services, making migration costly and complex  
     
   Limited Control: Developers have less access to the underlying infrastructure which may limit customization options

Model Answers for SOA

1. Service Oriented Architecture (SOA) is a design approach where applications are composed of loosely coupled services each encapsulating specific business logic.  
   Differences:  
   a) Monolithic Architectures: Applications are tightly coupled with all components interdependent   
   b) SOA: Promotes modularity, enabling independent development, deployment and scaling services
2. Service abstraction hides the internal implementation details of a service, exposing only its functionality through API’s  
   Importance:  
   a)Ensures security by restricting access to internal Logic   
   b)Simplifies integration and promotes loose coupling, allowing services to evolve independently
3. An ESB facilitates communication between services in an SOA environment  
   Functions:  
   a)Routing: Intelligent message rooting, including load balancing and failover  
   b)Message Mediation: Transforms and enhances messages between different protocols or formats  
   c)Centralized Management: Provides a single control point for monitoring and managing service interactions
4. Scalability: Services can be called independent based on demand, optimizing resource utilization  
     
   Flexibility: Supports integration of legacy systems and allows for agile adaptation to changing business requirements.
5. Complexity: Designing and managing an SOA system requires expertise in service integration and communication protocols  
     
   Performance: Service interactions over the Network may introduce latency and increase overhead  
     
   Security: Ensuring secure communication between distributed services requires robust policies

Lecture 5 (Microservices & SOAP & WDSL)

Microservices

Definition and Overview

Microservices

* Evolved from SOA, but with finer granularity and true decoupling
* Each microservice focuses on a single functionality, operates independently and avoid shared dependencies like ESB

Key Features

* Granularity: Services are small, independent and modular
* Decoupling: Microservices interact through lightweight protocols like REST, enabling independent updates and deployment
* Reusability: Services can be reused across applications or systems

RESTful Architecture

* Representational State Transfer (REST)
  + Architectural style based on HTTP
  + Resources identified by URI’s and accessed standard HTTP methods (GET, POST, PUT, DELETE)
  + Stateless communication improves scalability and performance

Benefits

* Agility: Supports CI/CD pipelines for faster deployment and updates
* Scalability: Fine- grained scaling for individual based on demand
* Resilience: Independent services reduce the impact of failures

Challenges

* Complexity: Requires robust service orchestration and monitoring
* Inter-Service Communication: Needs efficient protocols and API’s to maintain performance

SOAP

Basics of SOAP

* SOAP: Simple Object Access Protocol
  + Messaging protocol for exchanging structured information
  + Typically uses XML for message formatting and can operate over various protocols ( HTTP, SMTP)

Key Elements

* Envelope: Defines the start and end of the message
* Header: Contains metadata (security, transaction details)
* Body: Contains the main message payload
* Fault: Provides error handling details

Features

* Extensibility: Headers allow customization for security, routing etc.
* Reliability: Built in error handling mechanisms
* Protocol Agnostic: Not tied to HTTP, supports other transport protocols

WSDL

Definition and Purpose

* WDSL: Wen Services Description Language
  + XML-based language for describing web services
  + Specifies the location, methods and data formats of a service

Key Components

* Types: Define data types used in messages
* Message: Describe input/output data for operations
* PortType: Define operations and associated messages
* Binding: Specify the communication protocols and data formats
* Service: Group multiple ports to define endpoints

Benefits

* Interoperability: Standardized format allows integration access platforms
* Automation: Facilitates client-side stub generation for service interaction

Suggested Questions for Microservices

1. What are microservices and how do they differ from SOA
2. Explain the role of REST in microservices
3. Discuss the benefits of adopting a microservices architecture
4. What challenges arise when using microservices and how can they be addressed

Suggested Questions for SOAP

1. What is SOAP and what are its main components
2. Explain the role of the envelope and header in a SOAP message
3. Discuss the advantages of SOAP compared to REST
4. What are the challenges or limitations of using SOAP

Suggested Questions for WSDL

1. What is WSDL and why is it important in web services
2. Explain the core components of WSDL and their roles
3. How does WSDL support interoperability in web services
4. Discuss the relationship between WSDL and SOAP

Model Answers for Microservices

1. Microservices are an architectural style where applications are composed of small, autonomous services each focusing on a single functionality   
     
   Differences:  
   a)Granularity: Microservices are finer-grained than SOA services  
   b)Decoupling: Microservices minimize dependencies, avoiding shared resources like ESB used in SOA  
   c) Scalability: Individual microservices can scale independently whereas SOA services often scale together due to shared infrastructure
2. REST provides a lightweight communication mechanism for microservices  
   a)Uses HTTP methods (GET,POST,PUT,DELETE) for interactions   
   b)Resources are identified by URIs, ensuring simplicity and scalability  
   c)Stateless communication supports high scalability and fault tolerance
3. Agility: Enables continuous integration and delivery (CI/CD)  
   Scalability: Fine-grained scaling optimizes resource use  
   Resilience: Failure in one service doesn’t disrupt the entire application  
   Reusability: Modular services can be reused in different contexts
4. Complexity: Requires robust service orchestration tools (Kubernetes)  
   Inter-Service Communication: Efficient APIs and protocols like gRPC or REST are essential.  
   Data Management: Distributed data storage needs consistency mechanism  
   Monitoring: Tools like Prometheus and Grafana help maintain observability

Model answers for SOAP

1. SOAP is a protocol for exchanging structured information in distributed systems  
     
   Main Components:  
   a) Envelope: Defines the message structure   
   b) Header: Contains metadata like security tokens  
   c) Body: includes the actual message content  
   d) Fault: Handles errors and status information
2. Envelope: Marks the start and end of SOAP message and defines its schema  
   Header: Includes optional metadata for routing, authentication and transaction management
3. Protocol Independencies: SOAP supports HTTP,SMTP,FTP while REST is limited to HTTP  
   Reliability: Built in error handling and support for WS-Reliable Messaging.  
   Security: Advanced features like WS-security enable encryption and signing for end to end protection   
   ACID Transactions: WS-AtomicTransaction ensures transactional integrity
4. Complexity: SOAP messages are verbose due to XML sterilization  
   Performance: Processing large XML payloads can be slow  
   Tight Coupling: SOAP services often depend on WSDL making integration rigid

Model answers for WSDL

1. WSDL is an XML-based language used to describe web services, including their locations, operations and message formats  
     
   Importance:  
   a)Provides a standardized format for service descriptions  
   b) Enables automated generation of client-side code to interact with services
2. Types: Define the data types used in messages  
   Message: Describe input/output data for operations  
   PortType: Define the operations and associated messages  
   Binding: Specify protocols and formats for communication  
   Service: Group endpoints for accessing the service
3. Standardized definitions of data types and operations ensure compatibility across platforms   
     
   Allows integration between systems built on different technologies  
     
   Abstracts service implementation details, focusing on the interface
4. WSDL is often used to describe SOAP based web services  
     
   a)Specifies the structure of SOAP messages  
   b)Defines the transport protocols for SOAP communications  
   c)Acts as a contract, ensuring consistent service interaction

Lecture 7 (Containerization)

Overview of Containerization

* Containerization
  + Technology that packages software and its dependencies into lightweight, standalone containers
  + Provides OS-level isolation using Linux container (LXC) and Docker

Key Features

* Lightweight: Containers share the hist OS kernel, reducing overhead compared to virtual machines (VM’s)
* Portability: Docker containers can run consistently across various environments
* Isolation: Achieved through Linux namespaces and control groups

Docker Containers

* Docker images
  + Immutable, layered structure representing the application and its dependencies
  + Built using a DockerFile, similar to Heroku’s Buildpack
* Docker Daemon(dockerd)
  + Manages container lifecycles(build, run, stop)
* Docker Hub
  + Public registry for sharing and versioning Docker images
* Advantages
  + Full encapsulation of application and networking stack
  + Customizable port assignment and routing

Linux Containers (LXC)

* Resource limiting: Controls memory, CPU and network usage per container
* Prioritization: Allocates resources based on importance
* Accounting: Tracks resource usage for billing or benchmarking

LXC uses namespaces for isolation

* Pid: Isolates process IDs
* Net: Virtualizes networking interfaces
* Mnt: Provides separate views of mounted file systems

Benefits

* Efficiency: Reduced overhead compared to VMs
* Scalability: Containers start quickly and consume fewer resources
* Consistency: Ensures applications run identically across development, testing and production environment

Challenges

* Security: Containers share the host OS kernel, increasing the risk of kernel-level attacks
* Orchestration: Managing a large number of containers requires tools like Kubernetes
* Networking: Complex configurations for isolated and external communication

Suggested Questions for Containers

1. What is containerization and how does it differ from virtualization
2. Explain the role of Docker in containerization
3. Discuss the purpose and features of LXC in containerization
4. What are the benefits of containerization for application deployment
5. What challenges arise when using containers and how can they be mitigated

Model Answers for Containers

1. Containerization packages applications and their dependencies into lightweight, standalone units called containers  
     
   Differences:  
   a)Virtualization: Uses virtual machines (VMs) with dedicated guest Oss, creating significant overhead.  
   b) Containerization: Shares the host OS kernel, reducing resource consumption and improving efficiency.
2. Docker is a platform that simplifies the creation, deployment and management of containers  
     
   Key Features  
   a)Docker Image: Immutable, layered application packages built from DockerFile  
   b)Docker Daemon: Manages container lifecycle (build,run,stop)  
   c)Docker Hub: A public registry for sharing and versioning images  
   d) Portability: Ensures consistency across environments  
   e) Encapsulation: Includes all dependencies, runtime and libraries
3. Linux containers (LXC) provides OS-level virtualization using Linux kernel features like namespaces and cgroups  
     
   Namespaces:  
   a)pid: Isolates process IDs  
   b)net: Virtualizes networking interfaces for containers  
   c)mnt: Separates file system views for processes  
     
   Cgroups:  
   a)Controls resource allocation (CPU, memory, network)  
   b)Enables resource usage tracking and prioritization  
     
   Purpose: Ensures lightweight, secure isolation for processes running on the same OS
4. a)Efficiency: Containers share the host OS kernel, reducing overhead compared to VMs  
   b)Portability: Containers run consistently across development , testing and production environment.  
   c)Scalability: Containers start quickly and allow fine-grained resource scaling  
   d)Consistency: Encapsulation ensures predictable application behavior
5. Security:  
   a)Containers share the host OS kernel, increasing the risk of kernel level attacks  
   b) Mitigation: Use container runtime isolation tools like seccomp or AppArmor  
     
   Orchestration:  
   a) Managing many containers becomes complex  
   b)Mitigation: Use orchestration tools like Kubernetes for scheduling, scaling and monitoring   
     
   Networking:  
   a)Configuring isolated and external communication can be complete  
   b)Mitigation: Use network plugins and tools like Calico or Flannel for networking management

Lecture 8 (Kubernetes)

* Overview of Kubernetes
  + Kubernetes (K8s) is an open source platform for automating the deployment scaling and management of containerized applications
  + Key Features
    - Dynamic Scaling: Adjusts container resources based on demand
    - Self-Healing: Automatically restarts, replaces or reschedules failed containers
    - Rolling updates and Canary Deployment: Supports progressive updates to minimize risk

Cluster Components

* Nodes
  + Host environment (VMs) where containers are run
  + Managed by the Kubelet, which monitors resource usage, health and pod status
* Pods
  + Smallest deployable unit, representing one or more containers sharing the same resources (network, storage)
  + Designed to be ephemeral, dynamically created, destroyed or relocated

Workload Resources

* Define application behavior within the cluster and automate pod management
* Types
  + Deployments: For stateless applications; support rolling updates
  + StatefulSets: For stateful application requiring persistent storage and stable network identifies
  + Jobs: For batch processing or finite tasks

Services

* Abstract networking between pods and external clients
* ClusterIP: Provides a stable internal IP for pods within the cluster
* NodePort and LoadBalancer:
  + NodePort: Exposes services outside the cluster via a static port
  + Load Balancer: Distributes traffic across nodes using an external Load Balancer

Benefits

* Scalability: Handles, fluctuating workloads and ensures efficient resource utilization
* Resilience: Automates recovery from failures
* Portability: Supports containerized applications across different environments

Challenges

* Complexity: Requires expertise to configure and manage clusters
* Networking: Demands advanced understanding of pod to pod and external communication

Suggested Questions for Kubernetes

1. What is Kubernetes and why is it important in container orchestration
2. Explain the role of nodes and pods in a Kubernetes cluster
3. Discuss the different types of workload resources in Kubernetes and their use cases
4. What are the benefits of using Kubernetes for managing containerized applications
5. What challenges can arise with Kubernetes and how can they be mitigated

Model Answers for Kubernetes Questions

1. Kubernetes (K8s) is an open source platform that automates the deployment, scaling and management of containerized applications   
     
   Importance:  
   a)Orchestration: Ensures efficient allocation and management of containers across a cluster  
   b)Scalability: Dynamically adjusts resources to handle fluctuating workloads  
   c)Self-Healing: Automatically restarts , replaces or reschedules failed containers, ensuring application availability  
   d) Automated Rollouts and Rollbacks: Facilitates updates to applications while minimizing downtime and risk
2. Nodes:  
   a)Physical or virtual machines that run containers   
   b)Managed by the Kubelet, which ensures that pods are health and resources are used efficiently  
     
   Pods:  
   a)Smallest deployable unit in Kubernetes, representing one or more containers that share the same resources  
   b) Designed to be ephemeral and are dynamically created, deleted or relocated based on workload requirements  
   c) Containers in a pod share the same network namespace and can communicate with each other using localhost.
3. Workload resources define application behavior and automate pod management  
     
   Deployments:  
   a)Use case: Stateless applications(Web servers, RESTfull API’s)  
   b)Features: Supports rolling updates, replicas and fault tolerance  
     
   StatefulSets:  
   a)Use Case: Stateful applications requiring persistent storage (databases)  
   b) Features: Provides stable Network IDs, persistent storage and ensures pod order.  
     
   Jobs:  
   a)Use case: Batch processing or finite tasks(data migration, computations)  
   b)Features: Runs to completion, supports parallel execution and restarts failed pods
4. a)Scalability: Automatically scales resources based on workload demand  
   b)Self-Healing: Monitors container health and replaces failed containers  
   c)Portability: Runs on any infrastructure, from local machines to public clouds  
   d) Automation: Streamlines deployment, scaling and updates through declarative configurations (YAML files)  
   e) Resource Efficiency: Optimally schedules containers to maximize resource utilization
5. Complexity:  
   a) Challenge: Configuring and managing clusters requires expertise  
   b) Mitigation: Use managed Kubernetes services  
     
   Networking:  
   a) Challenge Managing pod to pod communication and external traffic is complex  
   b) Mitigation: Use network plugins and Kubernetes-native abstractions  
     
   Security  
   a) Challenge: Exposes risks due to shared infrastructure  
   b) Mitigation: Use role based access control (RBAC) pod security policies and network policies to enforce isolation

Lecture 11 (MVC\_IOC\_Beans & Servlets & Servlet Containers)

I don’t know if we actually have to know these things because these are more like coursework/exercise things !!!

Model View Controller (MVC)

Definition

* Architectural pattern dividing an application into three parts
  + Model: Manages application data
  + View: Handles UI logic
  + Controller: Links the model and view, processes requests and updates the view accordingly

Inversion of Control (IoC)

* Concept
  + Dependency Injection is used to manage dependencies between objects
  + IoC container
    - Creates objects and their dependencies
    - Injects dependencies at runtime without hardcoding them
  + Benefits
    - Reduces coupling
    - Improves reusability and testability

Beans in Spring Framework

* Definition
  + Objects managed by the Spring IoC container
* Annotations
  + Bean: Method- level annotation; explicitly defines beans
  + Component: Class- level annotation; auto- detectable by the IoC container
  + Specializations of Component
    - Service: Indicates business logic components
    - Controller: For MVC controllers
    - Repository: For Data Access Objects (DAOs)
  + Autowired: Automatically wires dependencies into fields, setters or constructors
* Bean Scopes
  + Singleton (default): Single instance per container
  + Prototype: New instance for every request
  + Web-specific scopes: Request, Session, Application and WebSocket

Servlets and Servlet Containers

* Servlets and Their Lifecycle
  + Definition
    - Server side Java programs that handle HTTP requests and generate responses
  + Lifecycle
    - Init(): Initilizes the servlet
    - Service():Processes requests and generates responses
    - Destroy(): Cleans up resources when the servlet is removed

Servlet Containers

* Provide runtime environments for servlets
* Responsibilities
  + Managing servlet lifecycle
  + Mapping HTTP requests to servlets
  + Facilitating communication between clients and servlets

JAX-RS and Jersey

* JAX-RS: Java API for RESTful Web Services
  + Uses annotations like Path, Get and Produces to define resources and responses
* Jersey: Reference implementation of JAX-RS
  + Includes deployment descriptors (<servlet>, <servlet mapping>) to configure servlets

Common Annotations

* Path
* GET,POST,PUT,DELETE
* Produces
* Consumes

Suggested Questions for MVC, IoC, Beans

1. Explain the Model View Controller (MVC) pattern and its components
2. What is inversion of Control (Ioc) and how does it improve application design
3. Describe the role of beans in the Spring Framework and differentiate between Bean and Component
4. Discuss the different bean scopes and their use cases in the Spring Framework
5. How does Autowired simplify dependency injection in Spring

Suggested Questions for Servlets and Servlet Containers

1. What is a servlet and what are its main lifecycle methods
2. Explain the role of a servlet container and its key responsibilities
3. What is JAX-RS and how does it support RESTful web services
4. Discuss the use of annotations like Path,GET and Produces in JAX-RS
5. How does jersey Implement Restufl services in a servlet container

Model Answers for MVC, IoC, Beans

1. Model: Manages applications data and business logic. Responsible for data processing and storage   
     
   View: Handles the presentation layer, displaying data from the model to the user  
     
   Controller: Processes user input, interacts with the model and updates the view accordingly  
     
   Purpose: MVC promotes separation of concerns , making applications easier to manage and scale
2. IoC is a design principle where the control of object creation and dependency injection is transferred from the application code to a container (IoC container)  
     
   Benefits:  
   a)Decoupling: Reduces dependencies between components  
   b)Reusability: Components can be reused across different contexts  
   c)Testability: Dependencies can be mocked or replaced during testing
3. Beans: Objects managed by the Spring IoC container  
   Bean: Method-level annotation that explicitly defines a bean in the configuration class  
   Component: Class level annotation for auto detecting and registering a class as a bean  
     
   Difference:  
     
   Bean is used for explicit control over bean creation   
   Component is used for classes that can be auto scanned by Spring
4. Singleton: Single instance per container  
    Use case: Stateless components  
   Prototype: New instance for every request  
    Use case: Stateful components  
   Request: One instance per HTTP request   
    Use case: Web request-specific beans  
   Session: One instance per HTTP session  
    Use case: User session data  
   Application: One instance per servlet context  
    Use case: Shared configuration across the application
5. Autowired automatically resolves dependencies by type and injects them into fields, setters or constructors  
   Reduces boilerplate code for creating and writing dependencies  
   Ensures dependencies are automatically provided by the IoC container, improving code readability and maintainability