CC Unit 1

20 February 2024 11:16 PM

Concepts and terminologies of Cloud computing

1. Cloud Computing:
   - Definition: Cloud computing refers to the delivery of computing services, including storage, processing power, and applications, over the internet.
2. Deployment Models:
   - Public Cloud:
     - Services are provided by third-party vendors and made available to the general public.
   - Private Cloud:
     - Cloud resources are used exclusively by a single organization.
   - Hybrid Cloud:
     - Combination of both public and private clouds, allowing data and applications to be shared between them.
3. Service Models:
   - Infrastructure as a Service (IaaS):
     - Provides virtualized computing resources over the internet.
   - Platform as a Service (PaaS):
     - Offers a platform allowing customers to develop, run, and manage applications without dealing with the complexity of infrastructure.
   - Software as a Service (SaaS):
     - Delivers software applications over the internet, eliminating the need for users to install, maintain, and run the applications locally.
4. Essential Characteristics:
   - On-Demand Self-Service:
     - Users can provision and manage computing resources as needed.
   - Broad Network Access:
     - Services are accessible over the network and can be accessed through standard mechanisms.
   - Resource Pooling:
     - Computing resources are pooled and shared among multiple users.
   - Rapid Elasticity:
     - Resources can be quickly scaled up or down to meet demand.
   - Measured Service:
     - Resource usage is monitored, controlled, and reported to provide transparency and billing.
5. Virtualization:
   - Enables multiple virtual instances of operating systems, servers, or networks to run on a single physical machine.
6. Containers:
   - Lightweight, portable, and self-sufficient units that can run applications and their dependencies, ensuring consistency across different environments.
7. Orchestration:
   - Automated arrangement, coordination, and management of computer systems, services, and resources.
8. API (Application Programming Interface):
   - Set of rules allowing different software applications to communicate with each other.
9. Scalability:
   - The ability of a system to handle increased workload by adding resources or nodes.

Cloud Delivery Models

1. Infrastructure as a Service (IaaS):
    - *Definition*: Provides virtualized computing resources over the internet.
    - *User Control*: Users have control over operating systems, storage, and networking.
    - *Example*: Amazon EC2, Microsoft Azure Virtual Machines.
2. Platform as a Service (PaaS):
    - *Definition*: Offers a platform with infrastructure, middleware, and development tools for application building.
    - *User Focus*: Users concentrate on application development; infrastructure is managed by the provider.
    - *Example*: Google App Engine, Heroku.
3. Software as a Service (SaaS):
    - *Definition*: Delivers software applications over the internet on a subscription basis.
    - *User Access*: Applications are accessed through a web browser without local installations.
    - *Example*: Google Workspace, Microsoft 365, Salesforce.

Cloud Deployment Models

Public Cloud:

- Significance:
    - Widely accessible to the general public, offering a scalable and cost-effective solution.
    - Suitable for organizations with variable workloads or those looking to minimize infrastructure investments.
- Benefits:
    - Cost-Efficiency: Pay-as-you-go model reduces upfront costs.
    - Scalability: Easily scale resources up or down based on demand.
    - Accessibility: Services are available over the internet from anywhere.
- Applicability:
    - Startups and small businesses with limited resources.
    - Projects with fluctuating or unpredictable workloads.

2. **Private Cloud:**

- Significance:
    - Provides dedicated and controlled resources for a single organization.
    - Offers enhanced security and customization options.
- Benefits:
    - Control: Full control over infrastructure and security policies.
    - Security: Ideal for industries with strict data privacy and compliance requirements.
    - Customization: Tailored to specific organizational needs.
- Applicability:
    - Enterprises with sensitive data and regulatory compliance requirements.
    - Organizations with predictable and steady workloads.

3. **Hybrid Cloud:**

- Significance:
    - Combines the benefits of both public and private clouds for flexibility.
    - Enables seamless data and application portability.
- Benefits:
    - Flexibility: Adapts to changing workload requirements.
    - Data Portability: Allows movement of data and applications between environments.
    - Cost Optimization: Balances cost-effectiveness and control.
- Applicability:
    - Enterprises with varying workloads and specific security needs.
    - Applications requiring both on-premises and cloud components.

Cloud Delivery models

1. Infrastructure as a Service (IaaS):

Theory:

- Provides virtualized computing resources over the internet.
- Users have control over the operating system, applications, and networking.
- Offers a scalable and flexible infrastructure.

Example: AWS EC2 (Elastic Compute Cloud)

- Users can provision virtual machines (EC2 instances) with specific configurations, install custom software, and have control over networking settings.

2. Platform as a Service (PaaS):

Theory:

- Offers a platform that allows customers to develop, run, and manage applications without dealing with the complexity of infrastructure.
- Provides a complete development and deployment environment.
- Abstracts underlying infrastructure, allowing developers to focus on coding.

Example: Heroku

- Developers deploy applications on Heroku without managing servers or infrastructure. Heroku handles scaling, deployment, and underlying platform management.

3. Software as a Service (SaaS):

Theory:

- Delivers software applications over the internet, eliminating the need for users to install, maintain, and run the applications locally.
- Users access applications through a web browser without worrying about backend infrastructure.
- Centralized maintenance and updates by the service provider.

Example: Salesforce

- Salesforce is a SaaS CRM platform where users access the CRM application through a web browser. Salesforce handles all infrastructure, maintenance, and updates centrally.

Horizontal Scaling:

1. Definition:
   - Also known as "scaling out," it involves adding more machines or nodes to a system to handle increased load.
2. Characteristics:
   - Distributes the load across multiple servers or instances.
   - Each server operates independently, and they share the overall processing load.
3. Advantages:
   - Improved fault tolerance as the failure of one server doesn't bring down the entire system.
   - Easier to add more capacity by adding new servers.
   - Cost-effective as it often involves using commodity hardware.
4. Use Cases:
   - Web applications, load balancers, and distributed databases benefit from horizontal scaling.

Vertical Scaling:

1. Definition:
   - Also known as "scaling up," it involves increasing the capacity of a single machine by adding more resources such as CPU, RAM, or storage.
2. Characteristics:
   - Involves upgrading the existing server to handle increased demand.
   - Resources are added vertically within the same machine or server.

3. Advantages:
    - Simplifies management as there's only one server to maintain.
    - Can be more cost-effective for certain workloads with consistent resource needs.
    - Suitable for applications that require more processing power within a single instance.
4. Use Cases:
    - Resource-intensive applications like large databases, complex algorithms, or scientific simulations may benefit from vertical scaling.

Cloud Enabling Technologies

Cloud-enabling technologies play a crucial role in the development, deployment, and management of cloud-based services. Here are some key cloud-enabling technologies presented in points:

1. Virtualization:
    - Description:
        - Abstracts physical hardware, allowing multiple virtual instances (VMs) to run on a single physical machine.
    - Benefits:
        - Efficient resource utilization, isolation of workloads, and flexibility in managing computing resources.
2. Containers:
    - Description:
        - Lightweight, portable, and self-sufficient units that package applications and their dependencies.
    - Benefits:
        - Consistent deployment across different environments, efficient resource utilization, and scalability.
3. Container Orchestration:
    - Description:
        - Automates the deployment, scaling, and management of containerized applications.
    - Examples:
        - Kubernetes, Docker Swarm, Apache Mesos.
    - Benefits:
        - Simplifies the management of containerized applications, ensures high availability, and automates scaling.
4. Microservices Architecture:
    - Description:
        - Organizes applications as a collection of small, independent, and loosely coupled services.
    - Benefits:
        - Scalability, ease of development, and maintenance, flexibility, and fault isolation.
5. Serverless Computing:
    - Description:
        - A cloud computing execution model where cloud providers automatically manage the infrastructure, and users pay for actual usage rather than pre-allocated resources.
    - Examples:
        - AWS Lambda, Azure Functions, Google Cloud Functions.
    - Benefits:
        - Cost efficiency, automatic scaling, and reduced operational overhead.
6. APIs (Application Programming Interfaces):
    - Description:
        - Set of rules and tools for building software applications, allowing them to communicate with each other.
    - Benefits:
        - Integration between different services, applications, or platforms.

Virtualization implementation 5 abstraction levels.

1. Application Level Virtualization:
    - Description:
        - Virtualizing individual applications, allowing them to run independently of the underlying system or other applications.
    - Key Features:
        - Applications operate in isolated environments, reducing conflicts and dependencies.
        - Facilitates easier deployment and management of applications.
2. Library Level Virtualization:
    - Description:
        - Library-level virtualization is not a common term. However, it might refer to the concept of dynamically linking libraries, allowing different applications to share common code libraries.
        - Alternatively, it could refer to application-level virtualization where libraries or dependencies are encapsulated with the application.
3. OS System Level Virtualization:
    - Description:
        - Virtualization at the operating system level, where multiple isolated instances (containers) share a single operating system kernel.
    - Key Features:
        - Containers encapsulate applications and their dependencies, providing lightweight and efficient resource utilization.
        - Examples include Docker and LXC (Linux Containers).
4. Hardware Abstraction Layer (HAL) Level:
    - Description:
        - Abstraction of hardware-specific details to allow software to run independently of the underlying hardware.
    - Key Features:
        - Provides a consistent interface for software to interact with hardware.
        - Enables portability of software across different hardware architectures.
5. Instruction Set Architecture (ISA) Level Virtualization:
    - Description:
        - Virtualization at the level of the CPU's instruction set architecture.
    - Key Features:
        - Allows the execution of multiple operating systems or applications on the same physical hardware.
        - Often implemented through hardware-assisted virtualization technologies like Intel VT-x or AMD-V.

Xen Hypervisor

1. Type of Hypervisor:
    - Type 1 Hypervisor:
        - Xen is a bare-metal hypervisor, meaning it runs directly on the hardware without the need for a host operating system.
2. Paravirtualization:
    - Description:
        - Xen employs paravirtualization, where the guest operating systems are modified to be aware of the virtualized environment.
    - Benefits:
        - Improved performance and efficiency as guest OSes actively participate in the virtualization process.
3. Hardware Virtualization Extensions:
    - Support:
        - Xen leverages hardware virtualization extensions such as Intel VT-x and AMD-V for improved performance and compatibility.
    - Benefits:
        - Enables running unmodified guest operating systems, not just paravirtualized ones.
4. Dom0 and DomU:
    - Dom0 (Domain 0):
        - The privileged domain running the Xen kernel, responsible for managing the hypervisor and controlling access to physical resources.
    - DomU (Domain U):
        - Unprivileged domains running guest operating systems.
5. Hypervisor Control Interface (XenStore):
    - Description:
        - A shared database used for communication between Dom0 and DomU.
    - Use:
        - Facilitates coordination and information exchange between different domains.
6. Xen Tools:
    - Description:
        - A set of utilities provided to enhance the management and interaction with Xen virtual machines.
    - Features:
        - Includes tools for managing VMs, configuring networking, and monitoring performance.