CC - 315325

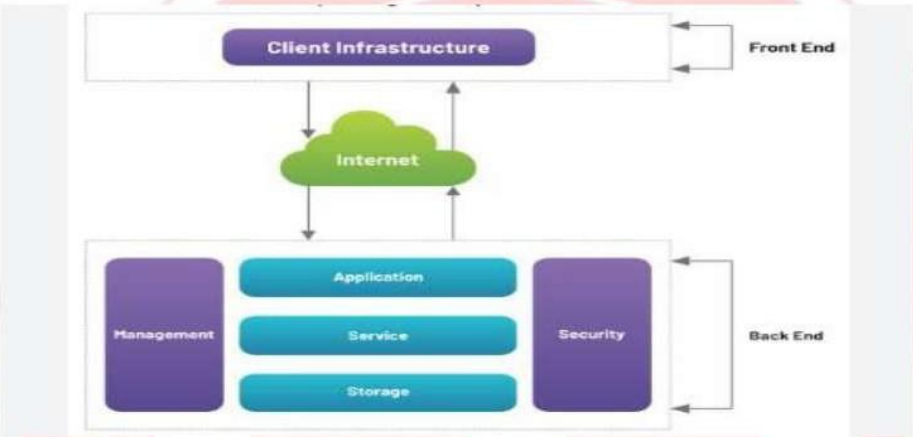

V2V EDTECH LLP
Online Coaching at an Affordable Price.
OUR SERVICES:
• Diploma in All Branches, All Subjects
• Degree in All Branches, All Subjects
• BSCIT / CS
• Professional Courses
+91 93260 50669
V2V EdTech LLP
v2vedtech.com
v2vedtech

Chapter 3 - Cloud Storage, Monitoring and Management

Introduction –

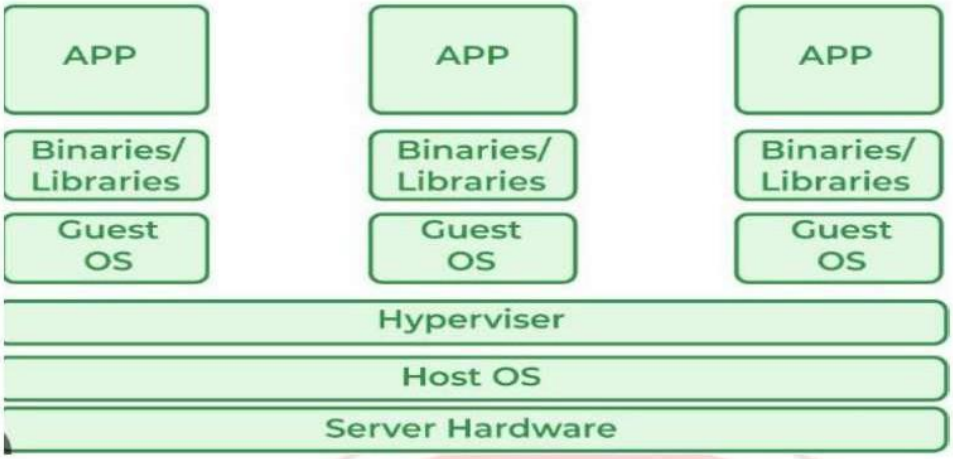
3.1 Cloud Storage System Architecture

cloud storage system architecture includes a front-end for user access, a back-end with virtualized servers and storage, and networking to connect them. It consists of components like a storage resource pool, storage virtualization software, and access interfaces (like APIs or web services) that allow users to interact with the stored data on demand. The architecture supports different storage types (object, block, file) and provides features like scalability, elasticity, and multi-tenancy.



3.2 Virtualize Data Centre (VDC) Architecture, VDC Environment, Server, Storage, Networking, Desktop and Application Virtualization techniques and benefits

A Virtual Data Center (VDC) uses virtualization to create a flexible, cloud-based IT infrastructure, abstracting physical servers, storage, networks, desktops, and applications to run as virtual instances. Key benefits include reduced hardware and operational costs, improved resource utilization, enhanced scalability and flexibility, simpler management through centralized tools, better disaster recovery capabilities, and increased agility.



VDC Architecture and Environment

- **Integrated Cloud Infrastructure:** A VDC is a pool of computing, storage, and networking resources delivered as a service.
- **Virtualization Layers:** It's a multilayered system where each component—servers, storage, networks, desktops, and applications—is virtualized.
- **Resource Pooling:** Physical resources are pooled and dynamically allocated to virtual instances, maximizing hardware utilization.
- **Cloud-Native:** VDCs often leverage cloud computing principles, providing resources as a service with a pay-as-you-go model.

Virtualization Techniques and Components

- **Server Virtualization:** Running multiple virtual servers (VMs) on a single physical server, allowing dynamic resource allocation and increased hardware efficiency.
- **Storage Virtualization:** Pooling various storage devices and managing them as a single logical unit, improving storage utilization and simplifying management.
- **Network Virtualization:** Creating virtual networks on top of physical hardware, enabling the interaction and secure communication of virtual servers and ensuring isolated data exchange.

- **Desktop Virtualization (VDI):** Separating desktop environments from physical devices, allowing users to access their desktops from any location or device, improving flexibility and management.
- **Application Virtualization:** Decoupling applications from their underlying operating systems, eliminating compatibility issues and allowing apps to run without being tied to specific OS requirements.

Benefits of VDC

- **Cost Efficiency:** Reduces hardware purchases, operational expenses, electricity consumption, and the need for specialized IT staff.
- **Scalability & Flexibility:** Quickly scales resources up or down to meet changing demands without needing to buy new physical hardware.
- **Improved Resource Utilization:** Maximizes the use of underlying hardware by running multiple virtual instances on fewer physical servers.
- **Enhanced Agility:** Accelerates application deployment and allows for faster provisioning of IT resources, crucial for a competitive business environment.
- **Simplified Management:** Centralized management tools provide a single pane of glass for administering and optimizing virtual resources.
- **Disaster Recovery & Business Continuity:** Simplifies data backups, snapshots, and the rapid redeployment of virtual machines, making disaster recovery more efficient.
- **Sustainability:** Lowers energy consumption and the carbon footprint by reducing the number of physical servers required.

3.3 Cloud File Systems: Google File System (GFS) : Components, Features, Advantages and Disadvantages and Hadoop Distributed File System (HDFS) :Terminologies like Heartbeat, Balancing and Replication, Features and Limitations

Google File System (GFS) is a proprietary, scalable, distributed file system developed by Google, designed to store and process vast amounts of data across clusters of inexpensive commodity hardware. It was created to handle Google's large-scale, data-intensive applications, such as its search engine, by providing fault tolerance, high performance, and high availability through the use of large file chunks and data replication.

Components:

- **GFS Clients:**

  Applications or programs that interact with the GFS to request file operations (read, write, append, etc.).
- **GFS Master Server:**

  The central coordinator of the GFS cluster. It manages metadata (namespace, access control, mapping of files to chunks and their locations), handles chunk lease management, and garbage collection.
- **GFS Chunk Servers:**
  Store data chunks (typically 64MB each) and serve read/write requests from clients, directed by the master. They also store replicas of chunks for fault tolerance.

Features:

- **Fault Tolerance:**

  Achieved through data replication (default 3 copies) and automatic recovery from component failures.
- **Large File Optimization:**

  Designed for handling large files (multi-GB) efficiently, optimizing for streaming reads and appends.
- **Atomic Record Appends:**

  Allows multiple clients to concurrently append data to a file atomically, ensuring data integrity.
- **Snapshotting:**
  Provides the ability to create point-in-time copies of files or directories.

Advantages:

- High throughput for large data sets.
- Scalability for petabyte-scale storage.
- Cost-effective due to use of commodity hardware.
- Strong consistency for control-plane operations.

Disadvantages:

- Not optimized for small files and random writes.
- Single Master server can be a bottleneck for metadata operations.
- Limited POSIX compliance.

Hadoop Distributed File System (HDFS)

Terminologies:

- **Heartbeat:**

  Periodic messages sent by DataNodes to the NameNode to signal their liveness and report block information.

- **Balancing:**

  The process of redistributing data blocks across DataNodes to ensure even disk space utilization within the cluster.

- **Replication:**
  The mechanism of creating multiple copies of data blocks and storing them on different DataNodes to ensure fault tolerance (default replication factor is 3).

Features:

- **Fault Tolerance and High Availability:** Achieved through data replication and automatic recovery from DataNode failures.
- **Scalability:** Horizontally scalable by adding more DataNodes to the cluster.
- **Cost-Efficiency:** Designed to run on commodity hardware, reducing infrastructure costs.

- **High Throughput for Large Data Sets:** Optimized for batch processing and high-throughput data access.
- **Streaming Data Access:** Optimized for large streaming reads.

Limitations:

- **Not suitable for low-latency data access:** Optimized for batch processing, not interactive applications requiring quick responses.
- **Cannot handle many small files efficiently:** Metadata overhead becomes significant with a large number of small files.
- **Limited support for random writes:** Designed primarily for write-once, read-many workloads.
- **Single NameNode:** The NameNode can be a single point of failure in older HDFS architectures (mitigated by High Availability features in newer versions).

3.4 Service Provider and users, An architecture of federated Cloud computing : Model and It's Explanation

In a federated cloud architecture, multiple autonomous cloud service providers collaborate to offer a unified cloud experience to users by sharing resources and interoperating their systems. Key components include the **Cloud Broker**, acting as an intermediary for services and pricing, and **Users**, who access these services through the broker or directly. The model relies on resource sharing and interoperability to achieve scalability, availability, and cost optimization for services like applications, platforms, and infrastructure.

Model and Components

A typical federated cloud architecture includes the following key components:

- **Service Providers:** Multiple independent cloud providers that make their infrastructure, platforms, and services available for federation.
- **Users:** The end-users, who could be individuals, organizations, or applications that consume the services offered by the federated cloud.
- **Cloud Broker:** A crucial intermediary that helps users find, select, and integrate services from different providers, ensuring they get the best value and meet their specific needs.
- **Cloud Exchange:** A mechanism that maps user demands to the services offered by different providers within the federation.
- **Cloud Coordinator:** A component responsible for assigning resources from various providers to meet user demands and manage the overall workload.

Explanation of the Model

1. **Autonomy and Interconnection:** Each cloud service provider in the federation remains an independent and autonomous entity but establishes connections with other providers to share resources.
2. **Unified Service Delivery:** The goal is to present a single, unified cloud environment to the users, abstracting away the complexities of interacting with multiple individual providers.
3. **Interoperability:** The providers' systems must be able to interoperate, allowing for smooth communication and resource sharing between their diverse infrastructures.
4. **Location Transparency:** Applications running in a federated cloud should be location-agnostic, meaning they don't need to know the specific cloud provider where their components are running.
5. **Resource Sharing and Demand Accommodation:** The federation allows providers to share resources to accommodate spikes in demand and to offer a broader range of services than any single provider could offer alone.

6. **User Interaction:** Users can interact with the federated cloud, typically through a cloud broker, to discover and access services based on criteria like cost, performance, and trust.

7. **Monitoring and SLA Management:** Continuous monitoring is essential to track service performance against Service Level Agreements (SLAs), with mechanisms in place to detect and address SLA violations.

In essence, federated cloud computing creates a "cloud of clouds" where providers collaborate while maintaining their independence, offering users increased availability, scalability, and optimized costs through resource pooling and shared management.

3.5 Service Level Agreement (SLA)

A Service Level Agreement (SLA) is a formal contract between a service provider and a customer that defines the specific services to be delivered, the agreed-upon performance metrics, and the consequences for failing to meet those standards. It clarifies expectations, sets measurable goals, and establishes accountability for both parties, ensuring transparency and providing security for the customer while guiding the provider.

Key Components of an SLA

- **Scope of Service**: Clearly outlines the services the provider is responsible for delivering.
- **Performance Metrics**: Details the key performance indicators (KPIs) by which the service's quality and performance will be measured, such as uptime, response times, and resolution times.
- **Responsibilities**: Specifies the obligations of both the service provider and the customer.
- **Service Level Objectives (SLOs)**: Measurable targets that define the desired level of service quality and performance.
- **Remedies and Penalties**: Explains what happens if the agreed-upon service levels are not met, which could include financial penalties or other remedies.

Why SLAs Are Important

- **Establishes Clear Expectations**: An SLA creates a shared understanding between the provider and customer, minimizing misunderstandings.
- **Ensures Accountability**: It holds the service provider accountable for delivering the promised level of service.
- **Provides Transparency**: For the customer, an SLA offers transparency regarding the services they are receiving and the standards they can expect.
- **Protects Both Parties**: An SLA protects both the customer and the provider by documenting terms and preventing claims of ignorance if issues arise.
- **Guides Service Provision**: It serves as a concrete guideline for the service provider, clearly delineating their responsibilities and boundaries.

3.5.1 SLA management: 5 Phases of SLA management like Feasibility, On-Boarding, Pre-production, Production and Termination

SLA management involves five phases: **Feasibility** (assessing the viability of an SLA), **On-Boarding** (defining and formalizing the SLA), **Pre-production** (preparing for the service delivery), **Production** (the actual service delivery and monitoring), and **Termination** (ending the agreement). Each phase builds on the success of the previous one, ensuring the service level agreement is properly established, maintained, and concluded to meet both the service provider's and the client's needs.

1. Feasibility

- **Purpose:** To determine if the service can be delivered according to the proposed service level agreements (SLAs).
- **Activities:**
  - Analyze the technical capabilities and resources required to meet the SLA's terms.
  - Assess the costs and potential benefits of the proposed service.

2. On-Boarding

- **Purpose:** To establish the framework and formalize the agreement.
- **Activities:**
  - Define clear, measurable, and achievable performance targets for the service.
  - Develop and negotiate the service level agreement (SLA) document, a contract outlining the responsibilities and expectations of both parties.

3. Pre-production

- **Purpose:** To prepare for the service's launch.
- **Activities:**
  - Implement the necessary infrastructure and technology for service delivery.
  - Train staff on their roles and responsibilities related to the SLA.
  - Set up systems for monitoring service performance against the agreed-upon metrics.

4. Production

- **Purpose:** To deliver the service and manage its ongoing performance.
- **Activities:**
  - Provide the service as outlined in the SLA.
  - Continuously monitor and track the service provider's performance to ensure compliance with the agreement.
  - Manage any issues or deviations from the SLA and implement corrective actions.

5. Termination

- **Purpose:** To formally end the service agreement.
- **Activities:**
  - Review the overall performance and outcome of the SLA.
  - Formally cancel the contract, ceasing the monitoring of the agreement's performance.

3.5.2 Types of SLA: Infrastructure SLA and Application SLA

Infrastructure and Application SLAs are two distinct types of Service Level Agreements that focus on different layers of a technology stack. The former covers the underlying hardware and network, while the latter addresses the performance and availability of the software running on top of that infrastructure.

Infrastructure SLA

An Infrastructure SLA defines the quality of service for the fundamental components that make up a system. It ensures that the core hardware and network are consistently available and perform as expected.

Key metrics typically included in an Infrastructure SLA:

- **Availability/Uptime:** The percentage of time a system, such as a server or network, is available and operational. For example, a 99.9% uptime guarantee for servers.
- **Response time/Latency:** The amount of time it takes for a request to travel between a client and the server.
- **Packet delivery:** The percentage of data packets that are successfully received compared to the total number sent.
- **Power and environmental conditions:** In a data center setting, this covers the maintenance of power supplies, climate control, and physical security.
- **Disaster recovery:** Specifies the time it will take for a service to be restored after a major outage (Mean Time to Recovery).

Application SLA

An Application SLA focuses on the performance and availability of a specific software application or service from the user's perspective. It measures the quality of the end-user experience, which depends on both the application itself and the underlying infrastructure.

Key metrics typically included in an Application SLA:

- **First-call resolution rate:** The percentage of customer issues resolved during their first interaction with a support team.
- **Error rate:** The frequency of errors that occur within the application, such as failed transactions or coding errors.
- **Transaction time:** The time it takes for a specific user action or transaction to be completed within the application.
- **Resolution time:** The average time it takes for an application-related issue, once logged, to be completely fixed.

3.5.3 Life cycle of SLA: 5 Phases like Contract Definition, Publishing and Discovery , Negotiation, Operationalization and De-commissioning

The SLA lifecycle involves phases of Defining the agreement (contract definition), Communicating it (publishing and discovery), Reaching consensus (negotiation), Executing the service (operationalization), and Ending the agreement (de-commissioning). This framework helps ensure that both the service provider and customer have clear, measurable, and agreed-upon expectations for the service, leading to successful service delivery and ongoing improvement.

1. Contract Definition

- Purpose: To clearly define the scope of the service, expectations, and the responsibilities of both the service provider and the customer.
- Key Activities:
  - Identifying specific services to be included.
  - Defining measurable performance metrics (Service Level Objectives or SLOs) such as uptime, response time, and turnaround times.
  - Outlining the penalties and rewards associated with meeting or failing to meet these metrics.

2. Publishing and Discovery

- Purpose: To make the SLA widely known and accessible to all relevant stakeholders.
- Key Activities:
  - Distributing the formal SLA document to both the service provider and customer teams.
  - Ensuring that all parties understand their roles and responsibilities as defined in the SLA.

3. Negotiation

- Purpose: To establish a formal, mutually agreeable contract between the provider and the customer.
- Key Activities:
  - Reviewing the proposed SLA and clarifying any ambiguous terms.
  - Discussing and agreeing on the service levels, metrics, and responsibilities.
  - Finalizing the SLA document.

4. Operationalization

- Purpose: To put the SLA into practice and ensure the agreed-upon services are delivered.
- Key Activities:
  - Implementing the necessary processes and tools to deliver the service according to the SLA.
  - Actively monitoring the defined metrics to track performance.
  - Regularly communicating with the customer to ensure satisfaction and address any issues.

5. De-commissioning

- Purpose: To formally end an SLA when the service is no longer needed, or a new agreement is established.
- Key Activities:
  - Conducting a final review to ensure all obligations under the current SLA have been met.
  - Archiving the SLA document.
  - Potentially initiating the next cycle for a new or revised SLA.

3.6 Cloud Service life cycle phases: Service planning, service creation, service operation and service termination

The cloud service lifecycle includes phases such as Service Planning (strategy, requirements), Service Creation (design, development, implementation), Service Operation (deployment, maintenance, monitoring), and Service Termination (retirement), encompassing the entire journey from idea to end-of-life. These phases ensure that cloud services are effectively delivered, managed, and continuously improved to meet business objectives.

1. Service Planning

- Service Strategy: Defines the vision and objectives for the cloud service, aligning it with overall business goals.
- Requirements Definition: Identifies the specific business and technical requirements the service must fulfill, including security and compliance needs.

2. Service Creation

- Service Design: Involves architecting the cloud service, designing the network, storage, compute, and application layers to meet requirements.
- Service Implementation: The process of configuring the designed components and building the cloud service.
- Testing: Thoroughly testing the service for functionality, performance, security, and compliance before deployment.

3. Service Operation

- Service Deployment: Making the service available to users by executing the deployment plan.
- Service Maintenance & Operation: Ongoing activities include monitoring performance and availability, managing incidents, and implementing changes.
- Continuous Service Improvement: A continuous loop of feedback and enhancements that connects to all other phases to ensure ongoing optimization and value delivery.

4. Service Termination

- Service Retirement: The process of phasing out the cloud service at the end of its useful life, including data archiving and secure disposal.

4.2.1 Policy and Organizational Risks

Policy and organizational risks are distinct categories of risk that can impact a business, but they are also deeply interconnected. Policy risks arise from external changes in laws or government regulations, while organizational risks stem from internal factors like culture, structure, and operational processes.

Policy risks

Policy risks are the uncertainties and potential for negative consequences that result from changes to the legal and political environment in which a business operates.

Examples

- **Regulatory non-compliance:** The introduction of new or stricter regulations, such as data privacy laws (e.g., GDPR), can impose substantial costs for compliance, fines for non-compliance, and reputational damage.
- **Political instability:** Operating in politically unstable countries can expose a company to risks like the nationalization of assets or abrupt policy changes that impact economic stability.
- **Shifting policy priorities:** A change in government can lead to reversals of policies that once benefited a company. For example, a new administration could withdraw subsidies for renewable energy, hurting investments in that sector.
- **Environmental policy changes:** Stricter environmental protection laws could increase operational costs for a manufacturing company or restrict its business activities entirely.

Mitigation strategies

- **Political and regulatory monitoring:** Businesses can invest in staying informed about new and evolving legislation and the political climate in the regions where they operate.
- **Diversification:** Diversifying a business across multiple regions or markets can reduce its exposure to policy changes in any single area.
- **Lobbying and public affairs:** Engaging with policymakers can help a company shape or influence legislation in its favor or at least anticipate upcoming changes.

- **Contingency planning:** For highly regulated or sensitive markets, businesses can develop scenario plans for how they will adapt to potential adverse regulatory changes.

Organizational risks

Organizational risks are internal threats that arise from a company's internal weaknesses in its policies, procedures, structure, or culture.

Examples

- **Ineffective internal policies:** Missing or poorly communicated internal policies on issues like cybersecurity, harassment, or expense reporting can lead to security breaches, legal action, and a toxic work environment.
- **Weak corporate governance:** Inappropriate board structures, inadequate internal controls, or poor communication can lead to flawed strategic decision-making and a lack of accountability.
- **Poor communication and collaboration:** When departments operate in silos, information flow is stifled. This can lead to a lack of agility, missed opportunities, and the inability to identify and address risks effectively.
- **Risk-averse culture:** A company culture that discourages risk-taking can prevent innovation and growth. It can also lead employees to ignore or fail to report potential problems for fear of repercussions.
- **Inadequate training and awareness:** Insufficient employee training on security best practices or risk management procedures can make a company vulnerable to both internal and external threats, such as phishing attacks or human error.

Mitigation strategies

- **Build a strong risk culture:** Embed risk awareness and accountability into the company's culture by defining and communicating roles and responsibilities for risk management across all levels.
- **Centralize risk management:** A central risk management function can provide a consistent framework for identifying, assessing, and monitoring risks, which helps break down communication silos.

- **Invest in employee training:** Implement regular, ongoing training for employees on key topics like cybersecurity, compliance, and ethical behavior. This ensures the workforce is equipped to handle risks effectively.
- **Establish robust governance frameworks:** Ensure clear governance structures and integrated control frameworks to manage compliance with multiple internal and external standards. This provides clarity and reduces redundant efforts.
- **Encourage dialogue and feedback:** Move away from a "rules-only" mindset to one that promotes open dialogue and discussion about potential risks. Create safe channels for employees to report concerns without fear of reprisal.