Cloud Application & Development Foundation

[BE SE Sixth Semester]

Nepal College of Information Technology
POKHARA UNIVERSITY

Unit IV: Data Management

4.1 Data Security, Data Location, Data Control

4.2 Scalability and Cloud Services

4.3 Large-Scale Data Processing

4.4 Databases and Data-stores in a Cloud Platform

4.5 Data Archival

Importance of Data Security in the Cloud

Why Data Security Matters:

- Cloud environments store sensitive data
  (e.g., personal, financial, intellectual property)
- Shared infrastructure increases risk of unauthorized access
- Regulatory compliance mandates robust security

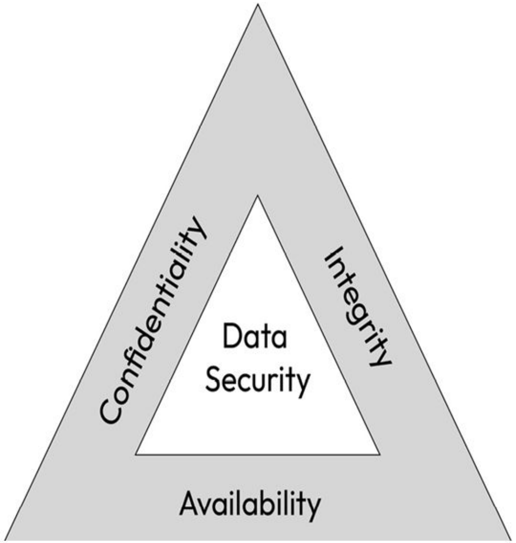
Key Challenges:

- **Multi-tenancy:** Multiple users sharing the same infrastructure
- **Data location:** Jurisdictional and compliance issues
- **Dynamic scalability:** Ensuring security during rapid scaling

Data Security in the Cloud

- Data security in the cloud involves protecting data stored, processed, or transmitted in cloud environments from unauthorized access, breaches, or loss.

- Increasing adoption of cloud services for sensitive data (e.g., financial, healthcare).
- Data security ensures confidentiality, integrity, and availability (CIA triad).
- Rising cyber threats: 43% of data breaches in 2024 involved cloud-based systems (IBM Security Report).

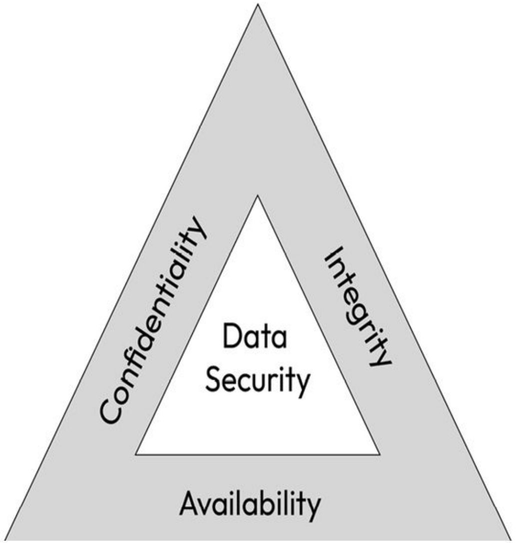


Core Principles of Cloud Data Security

Confidentiality:
- Ensuring data is accessible only to authorized users.
- Techniques: Encryption (at rest, in transit), access controls, secure key management

Integrity:
- Preventing unauthorized modifications to data.
- Methods: Hashing, checksums, audit logs.

Availability:
- Ensuring data is accessible when needed.
- Strategies: Redundancy, backups, DDoS protection.



Core Principles of Cloud Data Security

Authentication and Authorization:

- Identity verification (e.g., MFA, SSO)
- Role-based access control (RBAC) and attribute-based access control (ABAC)

Shared Responsibility Model:

- Cloud Provider: Secures infrastructure (e.g., physical servers, network).
- Customer: Manages data security, access policies, and application-level controls.

Key Standards: ISO 27001, NIST SP 800-53, SOC 2.

Common Threats to Cloud Data Security

- **Data Breaches**:Unauthorized access due to weak credentials or misconfigured services.
  Example: Exposed S3 buckets in AWS due to public access settings.
- **Data Loss**:Caused by accidental deletion, hardware failure, or ransomware.
  Mitigation: Regular backups, versioning, and disaster recovery plans.
- **Insider Threats**:Malicious or negligent actions by employees or contractors.
  Countermeasures: Role-based access control (RBAC), monitoring.
- **Insecure APIs**:Vulnerabilities in cloud service interfaces.
  Solution: API gateways, rate limiting, and authentication (e.g., OAuth).
- **Compliance Violations**: Failure to meet regulatory requirements
- **Account hijacking:** Compromised credentials
- **Distributed Denial of Service (DDoS):** Overwhelming cloud services

Key Strategies for Cloud Data Security

Encryption:

- Data at rest: AES-256 encryption for stored data.
- Data in transit: TLS/SSL for secure communication.
- Key Management: Use cloud-native solutions (e.g., AWS KMS, Azure Key Vault).

Access Control:

- Implement least privilege principle.
- Use Identity and Access Management (IAM) for fine-grained permissions.
- Multi-factor authentication (MFA) for enhanced security.

Network Security:

- Virtual Private Clouds (VPCs) for network isolation.
- Firewalls and intrusion detection systems (IDS).

Monitoring and Auditing:

- Tools: CloudTrail (AWS), Azure Monitor, Google Cloud Logging.
- Continuous monitoring for suspicious activities.

Data Masking and Tokenization:

- Protect sensitive data by obscuring it (e.g., replacing credit card numbers).

Cloud Security Guidelines

Best Practices:

- Implement end-to-end encryption for sensitive data
- Use strong identity and access management (IAM) policies
- Regularly update and patch cloud services
- Conduct employee training on security awareness
- Backup data with secure, offsite storage
- Data security is critical for trust and compliance in cloud environments
- Proactive measures and adherence to best practices mitigate risks effectively

Tools and Services:

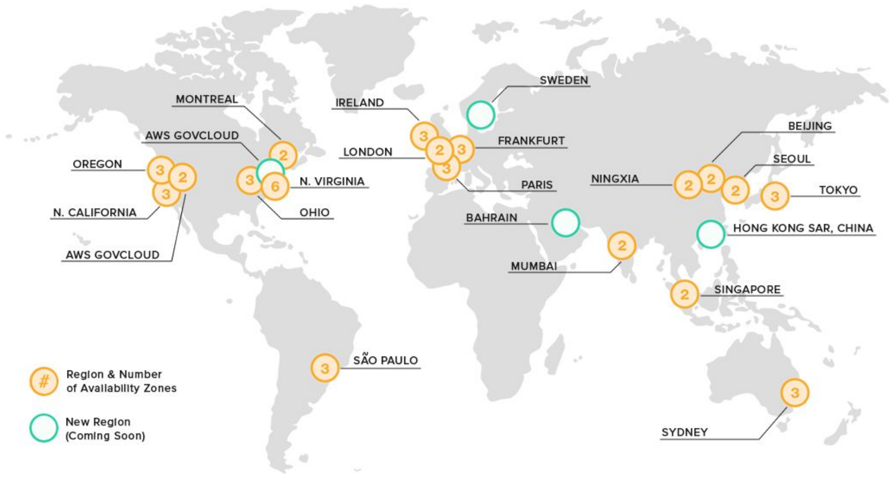
- AWS: IAM, KMS, GuardDuty
- Azure: Security Center, Key Vault
- Google Cloud: Cloud Armor, Identity-Aware Proxy

Compliance Requirements:

- GDPR: Data protection for EU citizens
- HIPAA: Safeguarding medical information
- PCI DSS: Securing payment card data
- SOC 2: Trust services criteria for security and privacy

Data Location in the Cloud

- Refers to the physical or logical geographic location where data is stored, processed, or transmitted in a cloud environment.
- Data location impacts legal and regulatory compliance.
- Example:
  Storing customer data in an EU data center to comply with GDPR.



Factors for selecting data location

	Compliance	*Latency*	*Cost*	*Services / Features*
Region 1	✓	15 ms	$$	✓
Region 2	✓	20 ms	$$$	X
Region 3	✓	80 ms	$	✓
Region 4	✓	15 ms	$$	✓
Region 5	✓	20 ms	$$$	X
Region 6	✓	15 ms	$	✓
Region 7	✓	80 ms	$	✓
Region 8	✓	15 ms	$	X

Factors Influencing Data Location

- **Performance**: Proximity to end-users reduces latency.

  Example: Content Delivery Networks (CDNs) like CloudFront.

- **Cost**: Data storage and transfer costs vary by region.

  Example: AWS pricing differences between US-East and Asia-Pacific.
- **Availability and Redundancy**: Multi-region setups for high availability.

  Example: Replicating data across multiple Availability Zones.
- **Vendor Lock-in**: Provider-specific regions may limit portability.
- **Security**: Physical and logical security measures vary by location.
- **Compliance**: Requirement to store data within national boundaries.

Managing Data Location Effectively

- **Region Selection**:
    - Choose regions based on user proximity, compliance, and cost.
    - *Example: Use AWS Region Selector tool.*
- **Data Replication**:
    - Multi-region replication for redundancy and performance.
    - *Example: Amazon S3 Cross-Region Replication.*
- **Geo-Restrictions**:
    - Implement access controls to restrict data movement.
    - *Example: AWS IAM policies with geographic conditions.*

Data Control in the Cloud

- Data control refers to the policies, processes, and technologies used to manage access, usage, and protection of data in cloud systems.
- Protects sensitive data from unauthorized access.
- Ensures compliance with legal and regulatory frameworks.
- Maintains trust in cloud-based services.
- Access Control: Who can access the data?
- Data Governance: Ensuring compliance with regulations (e.g., GDPR, HIPAA).
- Data Sovereignty: Managing data location and jurisdiction.

Guidelines of Data Control in the cloud

- Least Privilege: Grant minimal access rights necessary for tasks.
- Separation of Duties: Divide responsibilities to reduce risk of misuse.
- Data Segmentation: Isolate data based on sensitivity or purpose.:
- Identity and Access Management (IAM): Role-based access control (RBAC), multi-factor authentication (MFA).
- Encryption: Data encryption at rest and in transit (e.g., AES-256, TLS).
- Audit Trails: Logging and monitoring data access activities.

Practical Implementation and Tools

Implementing Data Control:

- Define clear data access policies aligned with organizational needs.
- Use cloud-native IAM tools to enforce policies.
- Integrate encryption and monitoring into workflows.

Popular Tools:

- AWS: IAM, AWS Key Management Service (KMS), CloudTrail.
- Azure: Azure Active Directory, Azure Key Vault, Security Center.
- Google Cloud: Cloud IAM, Cloud Audit Logs, Data Loss Prevention.

4.2 Scalability and Cloud Computing

- Ability of a system to handle increased loads without compromising performance
- Types: Vertical (scale-up: adding resources to a single node) and Horizontal (scale-out: adding more nodes)
- Supports dynamic workloads and unpredictable traffic spikes
- Ensures cost-efficiency and performance optimization
- Elasticity: Automatic scaling based on demand

Scalability Models in Cloud Services

Vertical Scaling (Scale-Up):

- Increase CPU, RAM, or storage on existing servers
- Pros: Simpler to implement, no architectural changes
- Cons: Limited by hardware capacity, potential downtime

Horizontal Scaling (Scale-Out):

- Add more servers to distribute workload
- Pros: Near-infinite scalability, fault tolerance
- Cons: Requires distributed system design, complex management

Cloud Service Support:

- IaaS: Virtual machines (e.g., EC2, Azure VMs)
- PaaS: App services (e.g., AWS Elastic Beanstalk, Google App Engine)
- SaaS: Built-in scalability (e.g., Salesforce, Google Workspace)

Cloud Services for Scalability

- Managed services reduce operational overhead
- Automatic scaling simplifies capacity management
- Support for diverse workloads (transactional, analytical)
- **AWS RDS:** Managed relational databases with automated scaling
- **AWS DynamoDB:** NoSQL database with seamless scalability
- **Azure Cosmos DB:** Globally distributed, multi-model database
- **Google BigQuery:** Serverless data warehouse for analytics

Auto-Scaling Databases

- Auto-scaling adapts to demand spikes and troughs
- Reduces manual intervention and over-provisioning
- Requires monitoring for performance and cost optimization
- Configuring Auto-Scaling
  - AWS DynamoDB Auto-Scaling: Adjusts read/write capacity based on utilization
  - Amazon Aurora Serverless: Scales compute/storage for relational workloads
  - Azure Cosmos DB: Throughput auto-scaling based on request units

Load Balancing for Data Services

Using Load Balancers

- Enhances availability by distributing traffic evenly
- Improves performance by reducing bottlenecks
- Integrates with auto-scaling for dynamic workloads
- AWS Elastic Load Balancer (ELB): Distributes database traffic
- Azure Load Balancer: Routes traffic for high availability
- Google Cloud Load Balancing: Global balancing for multi-region

Sharding and Partitioning

Sharding distributes data to handle large datasets

Partitioning improves read/write performance

Adds complexity in data management and queries

Techniques for Data Distribution

- DynamoDB Partitions: Divides data based on partition keys
- Azure Cosmos DB Sharding: Logical partitioning for global distribution
- Google Spanner: Automated sharding for relational databases

Caching for Scalability

Caching reduces database load for frequent queries

Improves read performance for repetitive access

Requires cache invalidation for data consistency

Using Caching Services

- AWS ElastiCache: In-memory caching with Redis/Memcached
- Azure Redis Cache: Managed Redis for low-latency access
- Google Cloud Memorystore: Managed in-memory data store

Serverless Data Services

Serverless eliminates infrastructure management

Scales automatically with workload demands

Cost-effective for sporadic/unpredictable workloads

Benefits of Serverless

- AWS Lambda with DynamoDB: Event-driven with auto-scaling
- Azure Functions: Integrates with Cosmos DB for workflows
- Google Cloud Functions: Works with BigQuery for analytics

Case Study: Scalable E-Commerce Platform

- Handles peak traffic (e.g., Black Friday)
- Combines caching, load balancing, auto-scaling
- Ensures high availability and low latency
- Example: AWS-Based E-Commerce Database
    - DynamoDB: Stores product catalog/user data with auto-scaling
    - ElastiCache: Caches product listings/user sessions
    - AWS ELB: Distributes traffic across read replicas

Challenges in Cloud Scalability

- Trade-offs between consistency, availability, partition tolerance
- Monitoring needed to balance performance and cost
- Design choices impact scalability success
- Key Challenges
    - Data Consistency: Eventual consistency in distributed systems
    - Latency: Increased in globally distributed databases
    - Cost Management: Auto-scaling can raise costs

4.3 Large-Scale Data Processing

- Large-scale data processing involves handling vast amounts of data efficiently using distributed systems.
- Powers big data applications like analytics, machine learning, and business intelligence.
- Cloud platforms enable scalable, cost-effective processing.
- Eliminates need for on-premises infrastructure.
- Scalability, cost-efficiency, and accessibility.

Characteristics of Large-Scale Data

The 5 Vs:

- **Volume**: Sheer size of data.
- **Velocity**: Speed of data generation (e.g., streaming data).
- **Variety**: Diverse formats (structured, unstructured, semi-structured).
- **Veracity**: Uncertainty and accuracy of data.
- **Value**: Extracting meaningful insights.

Examples:

- Social media posts, IoT sensor data, e-commerce transactions.

Cloud-Based Data Processing Frameworks

Popular Frameworks:

- Apache Hadoop: MapReduce for batch processing.
- Apache Spark: In-memory processing for speed.
- Google BigQuery: Serverless data warehousing.
- AWS Redshift: Managed data warehouse.

Key Features:

- Distributed storage (e.g., HDFS, S3).
- Parallel computation across clusters.
- Fault tolerance and scalability.

Batch Processing

- Processes large datasets in fixed-size batches.
- Examples: AWS Batch, Azure Data Factory, Google Dataflow.
- Common for ETL (Extract, transform, load) workflows.
- Ideal for non-time-sensitive tasks (e.g., nightly reports, data warehousing).
- High throughput, but not real-time.
- Workflow:

  Raw Data → Batch Job Scheduler → Processed Data → Storage

Stream Processing

- Processes data in real-time as it arrives.
- Examples: AWS Kinesis, Azure Stream Analytics, Google Pub/Sub.
- Used for live dashboards, fraud detection, IoT analytics
- Low-latency analytics for time-sensitive applications.
- Requires robust infrastructure to handle data velocity.

Distributed Computing Frameworks

- Frameworks for parallel processing: Apache Hadoop, Apache Spark.
- Cloud implementations: AWS EMR, Azure HDInsight, Google Dataproc.
- Handles massive datasets via distributed computing.
- Scales horizontally across clusters.
- Spark’s in-memory processing outperforms Hadoop MapReduce.

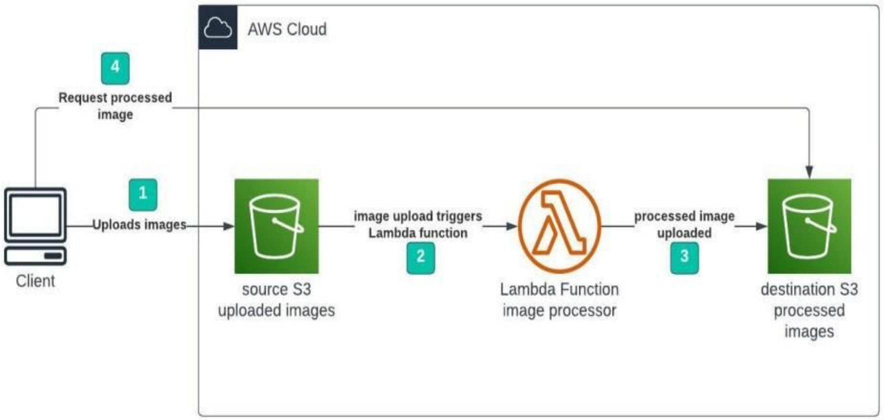
  Architecture of Spark on AWS EMR:

  [Master Node → Worker Nodes → Data Storage (S3)]

Amazon EMR *(Elastic MapReduce) is a cloud-based service from Amazon Web Services (AWS) that simplifies the processing of large datasets using frameworks like Apache Hadoop and Apache Spark*

Serverless Data Processing

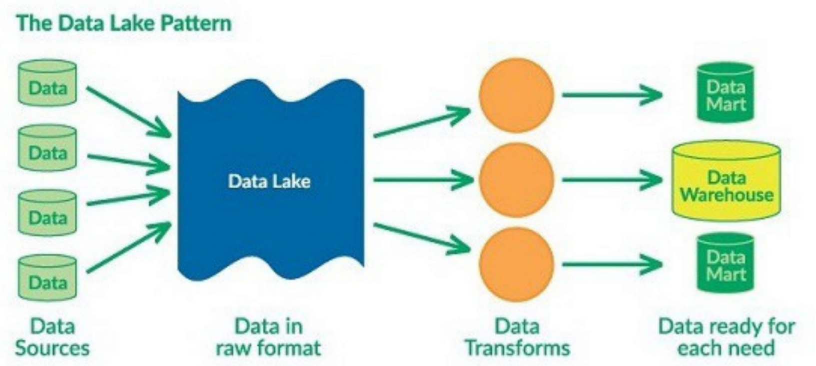
- Processes data using serverless platforms:
  AWS Lambda, Azure Functions, Google Cloud Functions.
- Triggered by events
  (e.g., Event Hubs, S3 uploads).
- Automatically scales with workload.
- Reduces operational overhead and costs.



Serverless Image Processor

Data Lakes

- Centralized repositories for raw, unstructured, and structured data.
- Examples:
  AWS S3 Data Lake, Azure Data Lake Storage, Google Cloud Storage.
- Supports batch, stream, and ML workloads.
- Flexible storage for diverse data types.
- Integrates with analytics and processing tools.



Performance Optimization

- **Partitioning**: Splits data into manageable chunks.
- **Compression**: Reduces storage and transfer costs.
- **Indexing**: Speeds up data retrieval.
- Optimization lowers costs and improves processing speed.
- Critical for large-scale systems.

- Example of partitioned data in S3:
  [s3://bucket/year=2025/month=06/day=25/]

Challenges in Large-Scale Processing

1. **Data Storage and Management**:

- Storing petabytes of data cost-effectively.
- Ensuring data durability and availability (e.g., 99.999999999% in S3).
- Managing data across regions for compliance

Solutions:

- Tiered storage (hot vs. cold storage).
- Data partitioning and indexing.
- Use of object storage (e.g., S3) vs. block storage.

2. **Data Transfer and Latency**

- Transferring large datasets to the cloud (e.g., 100 TB over the internet).
- Network latency in distributed systems.
- Bandwidth costs for cross-region data movement.

Solutions:

- Use physical transfer devices (e.g., AWS Snowball).
- Edge computing to process data locally.
- Caching and content delivery networks (CDNs).

Challenges in Large-Scale Processing

3. Scalability and Performance

- Scaling compute resources dynamically for variable workloads.
- Bottlenecks in distributed systems (e.g., shuffle operations in Spark).
- Resource contention in multi-tenant cloud environments.

Solutions:

- Auto-scaling policies in cloud platforms.
- Optimizing job configurations (e.g., partitioning, caching).
- Serverless architectures (e.g., AWS Lambda for event-driven tasks).

4. Cost Management

- Unpredictable costs due to pay-as-you-go pricing.
- Over-provisioning or under-provisioning resources.
- Hidden costs (e.g., data egress, API calls).

Solutions:

- Cost monitoring tools (e.g., AWS Cost Explorer).
- Reserved instances or savings plans for predictable workloads.
- Optimizing data formats

Challenges in Large-Scale Processing

5. Data Security and Privacy

- Protecting sensitive data in transit and at rest.
- Compliance with regulations (e.g., HIPAA, CCPA).
- Managing access control in shared cloud environments.

6. Complexity of Distributed Systems

- Designing and debugging distributed algorithms.
- Managing dependencies across services.
- Skill gap for developers new to cloud platforms.

Solutions:

- Use managed services (e.g., AWS Glue for ETL).
- Adopt DevOps practices (e.g., CI/CD, monitoring).
- Training and certifications (e.g., AWS Certified Big Data).

7. Fault Tolerance and Reliability

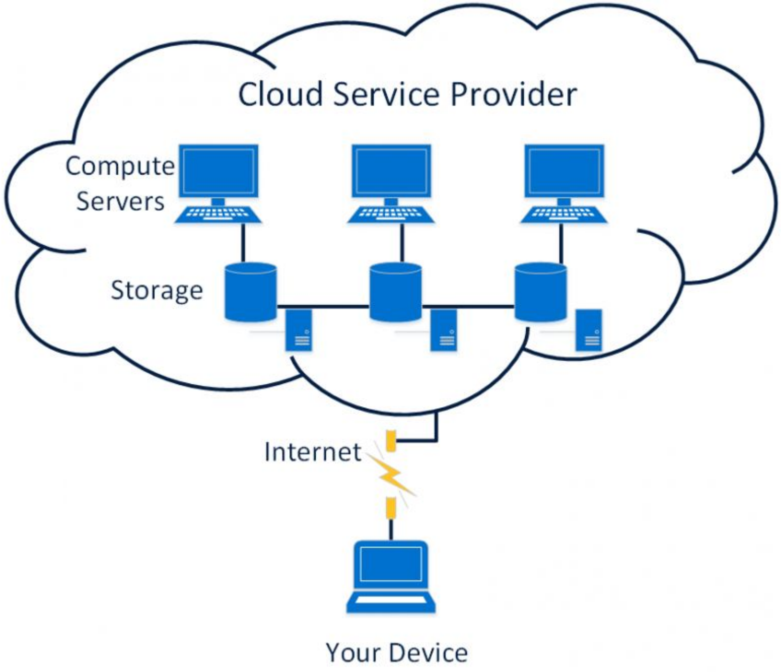
- Handling node failures in distributed systems.
- Ensuring job completion despite transient errors.
- Data consistency in distributed databases.

Solutions:

- Replication across availability zones.
- Checkpointing and retry mechanisms in frameworks like Spark.
- Use of eventual consistency models where applicable.

4.4 Cloud Databases and Data Stores

- Understand the role of databases and data stores in cloud platforms.
- Cloud databases/data stores manage structured, semi-structured, or unstructured data.
- Enable scalable, reliable data storage for applications
  (e.g., e-commerce, analytics).
- Cloud offers managed services, reducing maintenance overhead.
- Supports diverse workloads (transactional, analytical, real-time).



Issues in Cloud Databases

- **Cost Management:**
  Unoptimized queries or storage can increase costs.
- **Data Security:**
  Requires encryption, access controls.
- **Vendor Lock-In:**
  Proprietary services may limit portability.
- Proactive monitoring and design mitigate challenges.
- Balance performance, cost, and flexibility.

Relational Databases (RDBMS)

- Structured data storage using SQL (e.g., AWS RDS, Azure SQL Database, Google Cloud SQL).
- Supports MySQL, PostgreSQL, Oracle, etc.
- Used for transactional applications (e.g., banking, inventory).
- Strong consistency and ACID compliance.
- Managed services handle backups, scaling, and patching.

Primary Key Column

Product Table

ModelNumber	ProductName	ProductPrice	UnitCost
1111	Chair	$129.99	$65.00
1112	Desk	$599.99	$325.00
1113	Table	$299.99	$100.00
1114	File Cabinet	$159.99	$80.00

one-to-many relationship

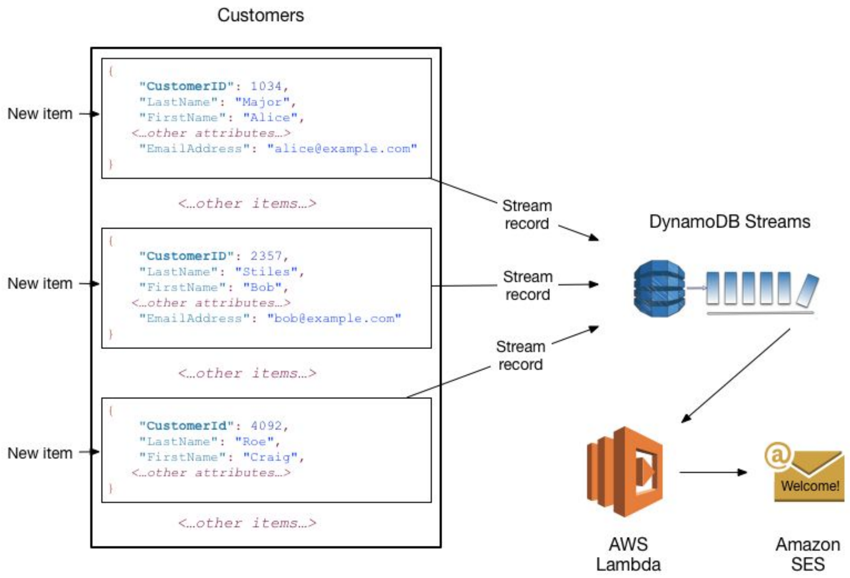
Sales Table

OrderNumber	ModelNumber	OrderQuantity	OrderDate
10-1256	1111	1	01/01/2021
10-1328	1111	3	02/15/2021
10-1524	1114	5	02/20/2021
10-1657	1113	1	03/02/2021

Foreign Key Column

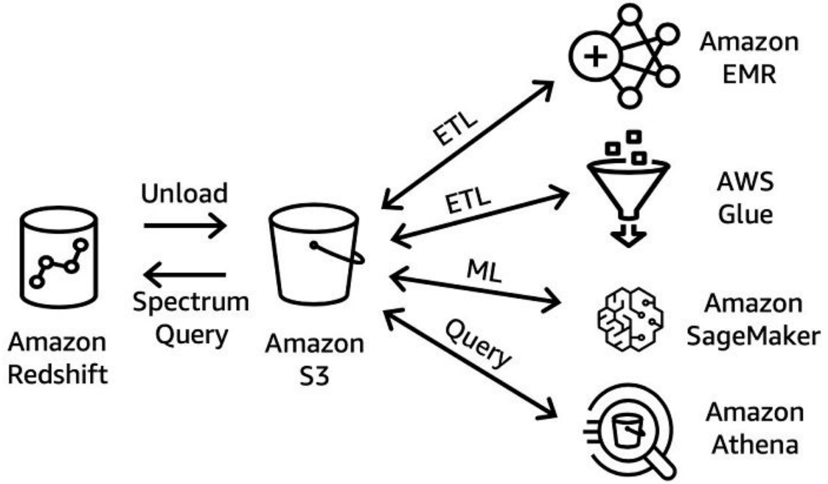
NoSQL Databases

- Handle semi-structured/unstructured data (e.g., AWS DynamoDB, Azure Cosmos DB, Google Firestore).
- Types: Key-value, document, column-family, graph.
- Used for high-scale, flexible applications (e.g., social media, IoT).
- Schema-less design for flexibility.
- Scales horizontally with low latency.



Data Warehouses

- Optimized for analytical queries on large datasets (e.g., AWS Redshift, Azure Synapse Analytics, Google BigQuery).
- Supports complex SQL queries for business intelligence.
- Columnar storage for fast analytics.
- Integrates with ETL and BI tools.
- Data warehouse workflow

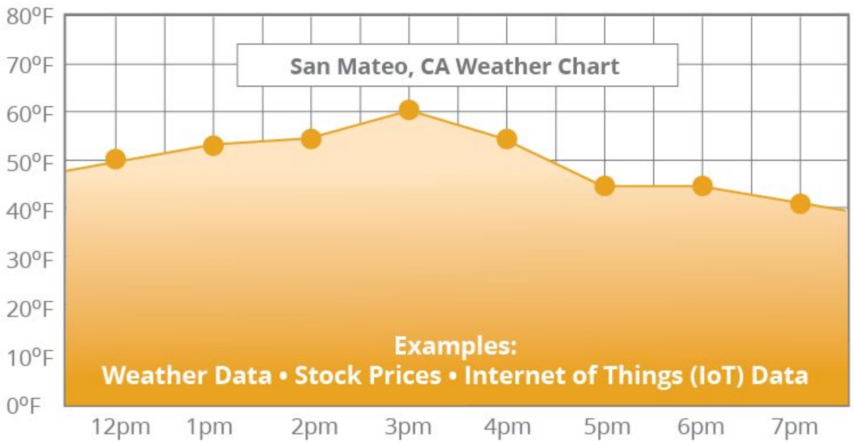


In-Memory Data Stores

- High-speed data access using RAM (e.g., AWS ElastiCache, Azure Cache for Redis, Google Memorystore).
- Used for caching, session management, real-time analytics.
- Sub-millisecond latency for read-heavy workloads.
- Complements databases by reducing load.

Time-Series Databases

- Optimized for time-stamped data (e.g., AWS Timestream, Azure Time Series Insights, Google Bigtable).
- Used for IoT, monitoring, and financial data.
- Efficient storage and querying of sequential data.
- Scales with high ingestion rates.



A Time Series Database is a database that contains data for each point in time.

Graph Databases

- Store and query relationships (e.g., AWS Neptune, Azure Cosmos DB Gremlin, Google Cloud Graph).
- Used for social networks, fraud detection, recommendation systems.
- Optimized for traversing complex relationships.
- Supports graph query languages like Gremlin, SPARQL.
  Graph example:
  Nodes (Users) → Edges (Follows) → Query Results

4.5 Data Archival

- Data archival involves storing infrequently accessed data for long-term retention.
- Ensures compliance, reduces costs, and preserves data for future use.
- Cloud provides scalable, cost-effective archival solutions.
- Balances accessibility with cost optimization.
- Store cold data at lower costs than active storage.
- Retain historical data for analytics or audits.
- Archival is critical for industries like healthcare, finance, and government.
- Reduces clutter in primary storage systems.

Cloud Archival Services

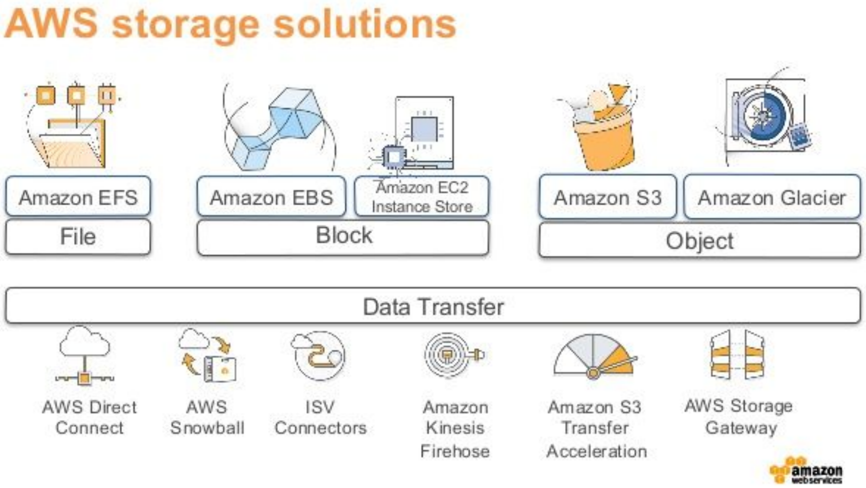
- Examples: AWS S3 Glacier, Azure Blob Archive, Google Cloud Archive.
- Features: Low-cost storage, tiered access, and durability (e.g., 99.999999999%).
- Use cases: Backup, disaster recovery, long-term retention.
- Managed services simplify archival processes.
- High durability ensures data integrity.

Storage Tiers for Archival

- **Standard Storage**: Frequent access (e.g., S3 Standard).
- **Infrequent Access**: Rare access (e.g., S3 Glacier).
- **Deep Archive**: Very rare access (e.g., S3 Glacier Deep Archive, Azure Cool/Archive).
- Choose tiers based on access frequency and retrieval needs.
- Deep archive offers lowest cost but longest retrieval time (hours to days).

Data Lifecycle Policies

- Automate transitions between storage tiers (e.g., AWS S3 Lifecycle Rules, Azure Blob Lifecycle).
- Example: Move data to Glacier after 30 days, Deep Archive after 180 days.
- Supports deletion of expired data.
- Lifecycle policies reduce manual effort and costs.
- Aligns storage with data usage patterns.



Storage Services (Summarized)

Amazon EBS

Block storage for use with Amazon EC2

VPC
EC2
EBS
BLOCK

Up to 16TB/volume
Up to 20K PIOPS
SSD backed
Cold & Throughput

Amazon EFS

Share File storage for use with Amazon EC2

VPC
EC2
EFS
FILE

Massively scalable Pay for what you use
High Performance
1000's of hosts

Amazon S3

Internet scale storage via API

VPC
OBJECT

Massively scalable storage & front-end
11 9's of durability
IA for infrequent access

Amazon Glacier

Storage for **archiving** and **backup**

OBJECT

$0.007/GB/month
11 9's of durability
Multiple copies across different DCs

AWS Storage Gateway

Integrates on-premises IT and AWS storage

S3, Glacier

Encryption

AWS Direct Connect
AWS Snowball
ISV Connectors
Amazon Kinesis Firehose
S3 VPC EndPoint
S3 Transfer Acceleration
Events
S3 Event Notifications
AWS Storage Gateway

amazon web services

Retrieval Options

- **Expedited**: Fast retrieval (minutes) for urgent needs (e.g., S3 Glacier Expedited).
- **Standard**: Balanced cost and speed (hours).
- **Bulk**: Low-cost, slower retrieval (days) for large datasets.
- Retrieval options balance cost and urgency.
- Plan retrieval strategy during archival design.
- Table: [Retrieval Type | Time | Cost | Use Case]

Unit IV: Data Management

4.1 Data Security, Data Location, Data Control

4.2 Scalability and Cloud Services

4.3 Large-Scale Data Processing

4.4 Databases and Data-stores in a Cloud Platform

4.5 Data Archival

Thank you