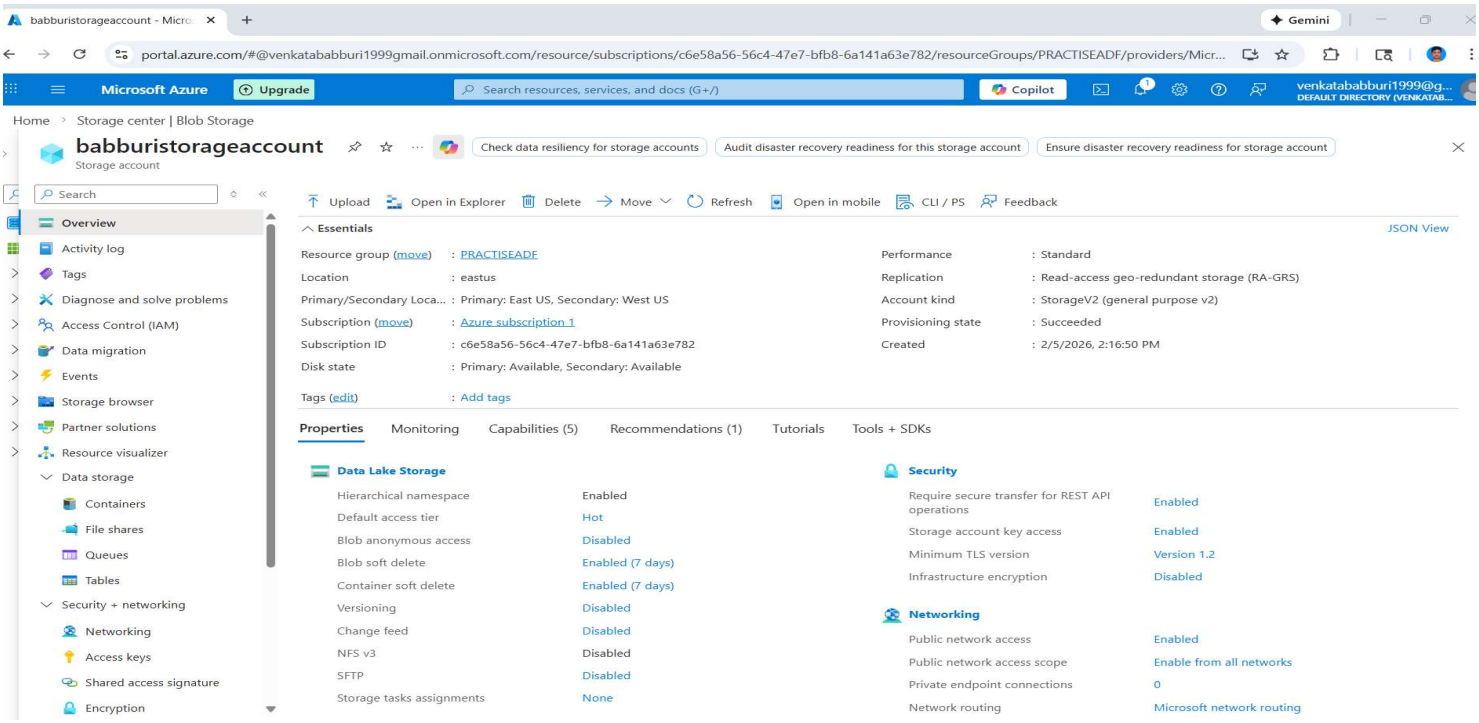
1. Azure Storage account

Monday, February 9, 2026 2:26 PM

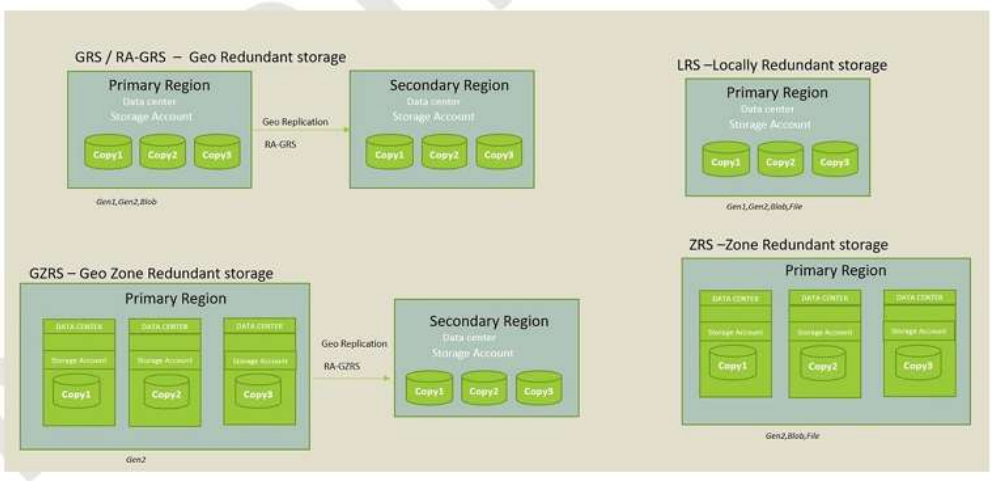
What is a Storage Account in Azure?

An **Azure Storage Account** is a service provided by Microsoft Azure that allows you to store and manage various types of data in the cloud. It acts as a container that provides a namespace for Azure Storage services and gives you access to Azure's scalable, secure, and durable storage options



Key Features of Azure Storage Accounts:

1. **Data Services:** An Azure Storage Account can hold several types of data services, including:
    - **Blob Storage:** For storing unstructured data such as images, videos, and documents. It's particularly useful for storing files that are accessed over the internet.
    - **File Storage:** Provides fully managed file shares in the cloud that are accessible via the Server Message Block (SMB) protocol. It's like having a network file share in the cloud.
    - **Queue Storage:** For storing large numbers of messages that can be accessed from anywhere via authenticated calls. It's often used for task scheduling and asynchronous processing.
    - **Table Storage:** A NoSQL key-value store for structured data. It's designed to handle large amounts of data with simple query operations.
    - **Disk Storage:** Provides persistent, durable, and high-performance storage for Azure Virtual Machines (VMs) in the form of managed disks.
2. **Scalability:** Azure Storage Accounts are designed to scale automatically to meet your needs. Whether you need to store a few gigabytes or several petabytes, Azure Storage can handle the load.
3. **Durability and Redundancy:** Azure provides multiple options for ensuring that your data is highly available and durable:
    - **Locally-Redundant Storage (LRS):** Copies your data three times within a single data center in a region.
    - **Zone-Redundant Storage (ZRS):** Copies your data synchronously across three Azure availability zones in a region.
    - **Geo-Redundant Storage (GRS):** Copies your data to a secondary region, hundreds of miles away from the primary location, for disaster recovery.
    - **Read-Access Geo-Redundant Storage (RA-GRS):** Provides read access to the data in the secondary region, in addition to the features of GRS.



4. **Security:**
   - **Encryption:** All data stored in Azure Storage is encrypted by default. Azure uses both server-side and client-side encryption to protect your data.
   - **Access Control:** You can control access to your storage account using Azure Active Directory (AAD) and shared access signatures (SAS), allowing you to define who can access your data and for how long.
   - **Firewall and Virtual Network Integration:** You can restrict access to your storage account to specific networks or IP ranges for additional security.
5. **Cost-Effectiveness:** Azure Storage offers different pricing tiers based on the type of storage, redundancy option, and access patterns (hot, cool, or archive tiers). This flexibility allows you to optimize costs based on your storage needs.
6. **Access Methods:** Data in an Azure Storage Account can be accessed via several methods:
   - **Azure Portal:** A graphical user interface for managing your storage account and its contents.
   - **Azure CLI/PowerShell:** Command-line tools for automating and managing your storage resources.
   - **Azure SDKs:** Software Development Kits (SDKs) for various programming languages (like .NET, Java, Python) that allow you to interact with Azure Storage programmatically.

   REST API: Provides direct access to Azure Storage services for custom integration

Types of Storage Accounts:

- **General-purpose v2 (GPv2):** Supports all the latest features and is the recommended type for most scenarios. It provides access to all Azure Storage services.
- **Blob Storage Account:** Specifically optimized for blob storage, with tiering options (hot, cool, and archive) to optimize cost based on access frequency.
- **File Storage Account:** Optimized for Azure Files, supporting premium file shares.
- **BlockBlobStorage Account:** Designed for workloads with high transaction rates or that require consistent, low-latency data access.

Use Cases:

- **Storing Files and Documents:** Store and access files, images, videos, and other unstructured data.
- **Backup and Restore:** Use Azure Storage as a backup destination for on-premises or cloud-based systems.
- **Disaster Recovery:** Ensure data availability with geo-redundant storage options.
- **Big Data Analytics:** Store large datasets for analysis with tools like Azure Data Lake Analytics or Azure Synapse Analytics.
- **Web and Mobile Applications:** Host and serve content such as web pages, videos, or static files directly from Azure Storage.

Encryption in Azure Storage Account

Azure Storage provides robust encryption features to protect your data at rest. This ensures that all your data is automatically encrypted before it is stored and decrypted before it is retrieved, without requiring any additional configuration or management from you.

🕐 **Microsoft-managed keys:** By default, Azure manages the encryption keys for you, which simplifies key management and ensures that your data is protected using Microsoft-managed keys.
🕐 **Customer-managed keys (CMK):** If you prefer more control, you can manage your own encryption keys using

Azure Key Vault. This gives you full control over the key lifecycle, including rotation and revocation. Customer-managed keys can also be used for auditing purposes, as you have visibility into the key usage.

Access Tiers in Azure Storage Account

Azure Storage offers different **access tiers** to help you optimize the cost of storing data based on how frequently the data is accessed. These access tiers allow you to store your data in a way that aligns with its usage patterns, enabling you to balance cost and performance.
Here's a breakdown of the Azure Storage access tiers:

1. Hot Tier:

- **Description:** The Hot tier is designed for data that is accessed frequently. It offers the lowest access latency and the highest throughput, making it ideal for data that needs to be accessed and processed regularly.
- **Use Cases:**
  - Active datasets, such as files and databases that are accessed frequently.
  - Content that is frequently updated or queried, like transaction logs.
  - Data for applications that require low latency access, such as web and mobile apps.
- **Cost:** Higher storage costs compared to the Cool and Archive tiers, but lower access costs.

2. Cool Tier:

- **Description:** The Cool tier is optimized for data that is infrequently accessed but needs to be stored for at least 30 days. It offers lower storage costs than the Hot tier but higher access costs.
- **Use Cases:**
  - Data that is not accessed frequently but still needs to be available for occasional access, like backups, archived data, or media content that is accessed seasonally.
  - Data that is stored for compliance or business continuity purposes.
- **Cost:** Lower storage costs than the Hot tier, but higher costs for data access and retrieval.

3. Archive Tier:

- **Description:** The Archive tier is intended for data that is rarely accessed and can tolerate higher retrieval times. This tier offers the lowest storage costs but the highest costs and latency for data retrieval.
- **Use Cases:**
  - Long-term archival data, such as compliance records, legal documents, or historical data that may be accessed once in a while.
  - Data that needs to be kept for extended periods but is not likely to be needed frequently.
- **Cost:** The lowest storage cost among all tiers, but the highest cost and latency for data access. Data in the Archive tier must be rehydrated (moved to the Hot or Cool tier) before it can be accessed.

Access Keys, SAS Tokens, and Why Azure Provides Two Keys (Key1 & Key2)

1. Access Keys:

- **What Are They?** Access keys are a pair of 512-bit keys generated by Azure for your storage account. These keys are used to authenticate and authorize access to your storage account's data services, including Blob Storage, Queue Storage, Table Storage, and File Storage.
- **Purpose:** Access keys allow full access to your storage account, including read, write, and delete operations across all data services. They are the primary means of programmatic access to Azure Storage services.
- **Usage:**
  - **SDKs and APIs:** When developing applications that need to interact with Azure Storage, you can use these keys to authenticate requests. The keys are included in the connection strings used by the Azure Storage SDKs or directly in API calls.
  - **Administrative Tools:** Tools like Azure Storage Explorer or custom scripts often use access keys to connect to and manage storage resources.

2. Shared Access Signature (SAS) Tokens:

- **What Are They?** A Shared Access Signature (SAS) token is a URI that grants restricted access rights to Azure Storage resources. Unlike access keys, which provide full access to a storage account, SAS tokens allow you to delegate access to specific resources with fine-grained control over permissions, duration, and IP restrictions.
- **Purpose:** SAS tokens are used to provide secure, temporary access to storage resources without sharing the full access keys. They are ideal for scenarios where you need to provide access to third parties or applications that should only have limited permissions.
- **Usage:**
  - **Time-Limited Access:** SAS tokens can be configured to expire after a certain period, ensuring that access is only available for a limited time.
  - **Permission Control:** You can specify which operations (read, write, delete, list, etc.) are allowed with the SAS token.
  - **IP Restrictions:** SAS tokens can be configured to only allow access from specific IP addresses or ranges, adding

an extra layer of security.
- **Resource-Specific Access:** SAS tokens can be scoped to a specific resource, such as a single blob or container, rather than the entire storage account.

3. Why Two Keys? (Key1 & Key2):

- **Key Rotation:** Azure provides two access keys (Key1 and Key2) for each storage account to facilitate key rotation, which is a best practice for security. Key rotation involves periodically changing your access keys to reduce the risk of compromise.
- **Purpose:** Having two keys allows you to rotate your keys without downtime. You can switch your applications from using one key to the other while you regenerate the first key. This ensures that your applications continue to function seamlessly during the key rotation process.
- **How It Works:**
  1. **Use Key1 for Authentication:** Initially, your application might use Key1 for accessing the storage account.
  2. **Regenerate Key2:** When you need to rotate keys, you can regenerate Key2.
  3. **Switch to Key2:** Update your application to start using Key2 for authentication.
  4. **Regenerate Key1:** Once the application is using Key2, you can safely regenerate Key1.
  5. **Switch Back if Needed:** You can continue this process to periodically rotate the keys.
- **Security Best Practice:** Regularly rotating access keys helps to minimize the risk if a key is accidentally exposed or compromised. By using the two-key system, you can ensure that your storage account remains secure while maintaining continuous access.

ADLS vs Blob Storage

Feature	Azure Data Lake Storage (ADLS)	Azure Blob Storage
Purpose	Optimized for big data analytics workloads	General-purpose object storage for unstructured data
Data Hierarchy	Supports hierarchical namespace (folders and subfolders)	Flat namespace; no native support for directories
Performance	Designed for high throughput and optimized performance for big data scenarios	High performance, but not specifically optimized for big data
Security	Granular access control with Azure Active Directory (AAD) integration	Role-based access control (RBAC) and shared access signatures (SAS)
Access Protocols	Supports Hadoop Distributed File System (HDFS) and POSIX-compliant ACLs	Standard REST APIs, SDKs, and tools like Azure Storage Explorer
Analytics Integration	Built-in support for analytics engines like Azure HDInsight, Azure Databricks, and Azure Synapse	Limited integration with big data analytics engines
Data Types	Optimized for structured, semi-structured, and unstructured data	Best for unstructured data (images, videos, documents, etc.)
Cost Structure	Higher cost, especially for hierarchical namespace features	Generally lower cost, suitable for simple storage needs
Use Cases	Big data analytics, complex data processing, enterprise-level data lakes	Backup, archiving, web applications, media storage

Azure File Share Service

Azure File Share is a fully managed cloud file storage service provided by Azure, which allows you to create and manage file shares that can be accessed via the Server Message Block (SMB) protocol or the Network File System (NFS) protocol. It is designed to replace or complement traditional on-premises file servers, providing scalable, secure, and accessible storage in the cloud.

Key Features of Azure File Share Service:

1. **Fully Managed Service:**
   - Azure File Share is a fully managed service, meaning Azure handles the underlying infrastructure, including hardware, networking, and software, allowing you to focus on your data and applications.
2. **SMB and NFS Protocols:**
   - Supports SMB protocol versions 2.1, 3.0, and 3.1.1, making it compatible with a wide range of Windows, Linux, and macOS operating systems.
   - NFS 3.0 protocol support is also available for Linux and Unix-based systems, which allows seamless integration with existing NFS-based workloads.
3. **Access from Anywhere:**
   - You can mount Azure file shares on your on-premises devices or in the cloud, providing seamless access to files from any location with an internet connection.
   - The service supports Azure AD integration for secure access and management.
4. **Scalability:**

- Azure File Shares can scale up to hundreds of terabytes, accommodating large volumes of data. You can create multiple file shares within a single storage account, each with its own capacity limit.

5. **Pricing Tiers:**
   - **Standard:** Designed for general-purpose file shares with cost-effective storage, supporting both SMB and NFS protocols.
   - **Premium:** Provides high-performance file shares optimized for low latency and high throughput, ideal for I/O-intensive workloads.

Use Cases for Azure File Share:

- **File Server Migration:** Migrate on-premises file servers to Azure to reduce infrastructure management overhead and improve accessibility.
- **Hybrid Cloud Solutions:** Use Azure File Sync to maintain a synchronized copy of your data on-premises and in Azure.
- **Application Storage:** Store configuration files, logs, and other application-related data that need to be shared across multiple instances or VMs.
- **Lift and Shift Applications:** For legacy applications that rely on SMB or NFS protocols, Azure File Shares offer a cloud-native solution without requiring application changes.
- **Persistent Storage for Containers:** Use Azure File Shares with Azure Kubernetes Service (AKS) to provide persistent storage for containerized applications.

How to Create and Use Azure File Shares:

1. **Create a Storage Account:**
   - In the Azure Portal, navigate to "Storage accounts" and create a new storage account.
2. **Create a File Share:**
   - Inside the storage account, go to "File shares" and click "Add" to create a new file share. Specify the name and quota for the file share.
3. **Mount the File Share:**
   - Use the connection string or script provided by Azure to mount the file share on your local machine or VM. This can be done using SMB or NFS protocols.
4. **Manage Files:**
   - Once mounted, you can manage files just like you would with a local file system, including creating, deleting, copying, and moving files.
5. **Use Azure File Sync (Optional):**
   - Install the Azure File Sync agent on a Windows Server and configure it to sync with your Azure File Share, enabling local caching and multi-site synchronization.

What is Azure Queue Storage?

Azure Queue Storage is a service provided by Microsoft Azure that allows you to store large numbers of messages that can be accessed from anywhere via authenticated calls. It is designed to help decouple components of a cloud application by enabling asynchronous communication between them. Azure Queues are particularly useful for handling tasks that do not require immediate processing or for distributing workloads across multiple processors.

Key Features of Azure Queue Storage:

1. **Message Storage:**
   - Each message in a queue can be up to 64 KB in size.
   - A queue can contain millions of messages, making it suitable for large-scale applications.
2. **Asynchronous Communication:**
   - Azure Queue Storage allows different parts of an application to communicate asynchronously. For example, a web application can place tasks in a queue, and background processes can process them later.
   - This decoupling enables better scalability and fault tolerance.
3. **Visibility Timeout:**
   - When a message is retrieved from a queue, it becomes invisible to other clients for a specified period (known as the visibility timeout). If the message is processed within this time, it can be deleted. If not, it becomes visible again for another process to handle.
   - This feature ensures that messages are processed at least once, even if a worker fails after retrieving a message.
4. **Poison Messages Handling:**
   - If a message repeatedly fails to process and exceeds a certain threshold of visibility requests, it is often moved to a "poison message" queue for further investigation. This prevents a single problematic message from blocking the processing of other messages.
5. **Time-to-Live (TTL):**
   - Messages in a queue can have a time-to-live (TTL) setting. If a message is not retrieved within the TTL, it is automatically deleted. By default, messages remain in the queue until they are explicitly deleted.
6. **Scalability:**
   - Azure Queue Storage is highly scalable, able to handle vast numbers of messages across distributed components of an application.
7. **Integration with Other Azure Services:**

- Azure Queue Storage can integrate seamlessly with other Azure services like Azure Functions, Azure Logic Apps, and Azure WebJobs. For instance, an Azure Function can be triggered to execute whenever a new message is added to a queue.
- This integration makes it easy to automate processing workflows based on queue messages.

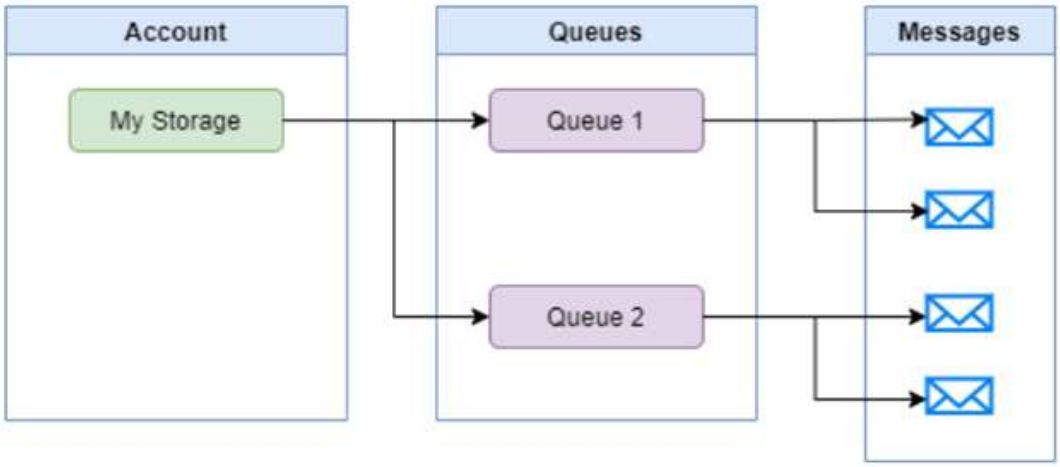
8. **Cost-Effective:**
   - Queue Storage is a cost-effective solution for building scalable, decoupled, and distributed applications. You pay for the storage used by the messages and the operations performed on the queue (such as adding and retrieving messages).

Use Cases for Azure Queue Storage:

1. **Task Scheduling:**
   - Queue Storage is often used for scheduling tasks that do not need to be executed immediately. For example, an e-commerce site might use a queue to process orders asynchronously after a customer completes a purchase.
2. **Workload Distribution:**
   - Azure Queues can distribute tasks across multiple processing nodes. For example, a queue can be used to distribute image processing tasks across multiple virtual machines, ensuring that each VM processes a part of the workload.
3. **Event-Driven Architectures:**
   - In an event-driven architecture, different parts of the system might generate events that need to be processed asynchronously. Queue Storage can be used to manage these events, ensuring that they are processed in the correct order.
4. **Decoupling Components:**
   - By using queues, different parts of an application can be decoupled, making the system more modular and easier to scale. For example, a front-end web application might place user requests in a queue, which are then processed by a separate backend service.

How to Use Azure Queue Storage:

1. **Create a Queue:**
   - In the Azure Portal, navigate to your storage account, and under "Queues," you can create a new queue by providing a unique name.
2. **Add Messages to the Queue:**
   - You can add messages to the queue using the Azure Portal, Azure SDKs, or REST API. Each message can contain text data up to 64 KB in size.
3. **Retrieve and Process Messages:**
   - Your application can retrieve messages from the queue, process them, and then delete them from the queue. If the processing fails, the message will become visible again after the visibility timeout.
4. **Monitor and Manage the Queue:**
   - Azure provides tools to monitor the number of messages in a queue, the rate at which messages are being added and processed, and other key metrics.



What is a NoSQL Database?

A **NoSQL database** is a type of database that provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in traditional relational databases (SQL databases). NoSQL databases are designed to handle large volumes of unstructured, semi-structured, or structured data, and are optimized for performance, flexibility, and scalability.

Key Characteristics of NoSQL Databases:

1. **Non-Relational Data Model:**
   - Unlike SQL databases, which use tables with rows and columns to store data, NoSQL databases use various data models, such as document, key-value, wide-column, and graph models.
2. **Scalability:**

    - NoSQL databases are designed to scale out horizontally, meaning you can add more servers or nodes to handle increased load, making them suitable for large-scale applications and big data scenarios.
3. **Flexible Schema:**
    - NoSQL databases often have a flexible schema, allowing you to store data without a predefined structure. This means you can easily add new fields or attributes to your data without having to alter the entire database schema.
4. **Distributed Architecture:**
    - Many NoSQL databases are distributed, meaning data is stored across multiple servers or nodes, which improves availability, fault tolerance, and scalability.
5. **High Performance:**
    - NoSQL databases are optimized for high read and write performance, making them suitable for real-time applications, such as social media platforms, online gaming, and IoT systems.

Types of NoSQL Databases:

1. **Document Databases:**
    - **Description:** Store data as documents, usually in JSON or BSON format. Each document is a self-contained unit of data that can have a different structure from other documents in the same collection.
    - **Examples:** MongoDB, CouchDB, RavenDB.
    - **Use Cases:** Content management systems, catalogs, user profiles, and real-time analytics.
2. **Key-Value Stores:**
    - **Description:** Store data as key-value pairs, where each key is unique and maps to a value. This is the simplest form of NoSQL database.
    - **Examples:** Redis, DynamoDB, Riak, Azure Table Storage.
    - **Use Cases:** Caching, session management, and real-time data processing.
3. **Wide-Column Stores:**
    - **Description:** Store data in tables, rows, and dynamic columns, where each row can have a different set of columns. This model is suitable for handling large datasets with high write throughput.
    - **Examples:** Cassandra, HBase, ScyllaDB.
    - **Use Cases:** Time-series data, sensor data, and large-scale analytical workloads.
4. **Graph Databases:**
    - **Description:** Store data in nodes and edges, where nodes represent entities, and edges represent the relationships between them. This model is highly effective for querying complex relationships in data.
    - **Examples:** Neo4j, Amazon Neptune, ArangoDB.
    - **Use Cases:** Social networks, recommendation engines, fraud detection, and network analysis.

Advantages of NoSQL Databases:

- **Scalability:** Easily scale horizontally to accommodate large volumes of data and high transaction rates.
- **Flexibility:** Store unstructured, semi-structured, or structured data without requiring a rigid schema.
- **Performance:** Optimized for fast reads and writes, making them ideal for real-time applications.
- **Distributed and Fault-Tolerant:** Often designed to run on distributed systems, ensuring high availability and fault tolerance.

Disadvantages of NoSQL Databases:

- **Lack of ACID Transactions:** Many NoSQL databases do not provide strong consistency guarantees (ACID transactions), which can be a disadvantage for applications requiring strict data consistency.
- **Complexity:** NoSQL databases may require more complex data modeling and querying, especially for developers accustomed to SQL databases.
- **Limited Support for Complex Queries:** While NoSQL databases excel in performance and scalability, they may not offer the advanced query capabilities found in SQL databases.

Use Cases for NoSQL Databases:

- **Big Data and Analytics:** Handling large volumes of diverse data for analytics, such as in IoT applications or social media platforms.
- **Content Management:** Storing and retrieving unstructured data, such as documents, images, and videos.
- **Real-Time Applications:** Supporting high-performance, real-time applications like online gaming, chat applications, and real-time analytics.
- **Graph-Based Queries:** Managing data with complex relationships, such as social networks, recommendation systems, and fraud detection.